Títol: *Interfície Bluetooth entre microcomputador i telèfon mòbil per a la captura i tractament de dades*

Volum: *1/1*

Alumne: *Jose Carlos Yáñez Nieto*

Director/Ponent: *Joan Climent Vilaró*

Departament: *ESAII*

Data: *26 de Gener de 2010*

DADES DEL PROJECTE

Títol del Projecte: Interfície Bluetooth entre microcomputador i telèfon mòbil per a la captura i tractament de dades

Nom de l'estudiant: Jose Carlos Yáñez Nieto
Titulació: Enginyeria tècnica en Informàtica de Sistemes
Crèdits: 22.5
Director/Ponent: Joan Climent Vilaró
Departament: ESAII

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Pere Marés Martí

Vocal: Gladys Miriam Utrera Iglesias

Secretari: Joan Climent Vilaró

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

ÍNDEX

CAPÍTOL 1

INTRODUCCIÓ

Aquest projecte final de carrera és una de les parts d'un projecte comercial consistent en integrar una sèrie de microcontroladors en una bicicleta elèctrica. L'objectiu és poder controlar l'esforç físic que realitza l'usuari de la bicicleta per així poder ajustar la força que fa el motor elèctric o per poder avaluar una possible evolució del seu estat físic. En el cas d'aquest projecte a més s'afegirà el control de les pulsacions cardíaques.

En el desenvolupament d'aquest projecte treballen diversos departaments i els sensors que havien de captar la força realitzada per l'usuari no estaven encara preparats, així que es va optar per realitzar el projecte emulant aquests sensors. L'eina escollida és la placa de desenvolupament utilitzada pels estudiants de l'assignatura SDMI de la Facultat d'Informàtica de Barcelona. Aquesta placa incorpora diversos elements com generadors de freqüència variable, generadors analògics, diversos canals de comunicació, sortides i entrades digitals entre molts altres, que estan connectats a un microcontrolador. Aquests elements ens permetran emular els sensors que envien les dades al microcontrolador, responsable de captar-les.

Però l'abast del projecte no acaba en generar i captar les dades, en aquest treball també s'inclou el tractament de les mateixes. Un cop el microcontrolador obté les dades provinents dels sensors de la bicicleta els enviarà a un dispositiu de l'usuari, que pot ser el seu propi telèfon mòbil. Es desenvoluparà l'aplicació per tal que el telèfon mòbil rebi i ofereixi a l'usuari un tractament de les dades que s'han generat durant la realització de l'exercici. La possibilitat d'obtenir els resultats en temps real a la pantalla del telèfon mòbil fa que l'usuari pugui conèixer en tot moment els paràmetres que s'estan mesurant. A més, les dades quedaran enregistrades per poder ser estudiades o consultades més tard per l'usuari o per exemple, per algun entrenador personal per comprovar els avenços o l'estat de salut de la persona que ha fet l'exercici.

1.1 OBJECTIUS DEL PROJECTE

L'objectiu principal d'aquest projecte és comunicar una bicicleta elèctrica equipada amb una sèrie de sensors amb un dispositiu mòbil amb la finalitat de tractar les dades captades provinents d'aquests sensors.

Utilitzar la placa de desenvolupament per emular, amb els seus elements, els sensors que portaria la bicicleta elèctrica. L'emulació ha d'ajustar-se en lo possible a la realitat, ja que la intenció és substituir en un futur els sensors emulats per sensors reals connectats a algun dels microcontroladors de la bicicleta.

A més d'utilitzar els elements de la placa de desenvolupament per emular els sensors, s'ha de captar els valors que aquests generen. Per aquest objectiu cal desenvolupar el firmware[3] per al microcontrolador. El firmware ha de ser robust, senzill i que, amb pocs recursos, sigui capaç d'oferir un bon rendiment.

Un altre objectiu és crear un sistema de tipus client – servidor entre el telèfon mòbil i la bicicleta. La bicicleta estarà esperant rebre peticions de connexió de dispositius per enviar les dades que generin els sensors. El dispositiu mòbil serà un client que es connectarà al servidor per rebre les dades i quan finalitzi l'exercici es desconnectarà.

La connexió entre la bicicleta i el telèfon mòbil s'ha de realitzar de forma senzilla, ja que molts usuaris potser poden trobar dificultats amb les noves tecnologies, com persones de la tercera edat. La transmissió de les dades s'ha de fer en temps real perquè pugui ser útil a l'usuari mentre realitza l'exercici. Aquesta connexió es farà a través de Bluetooth[1].

L'aplicació del telèfon mòbil ha d'oferir la possibilitat de rebre dades des de la bicicleta i ha de generar un arxiu de log al telèfon de l'usuari d'aquestes dades rebudes de la bicicleta. Aquest arxiu ha de poder ser descarregable i llegible des d'un PC qualsevol sense necessitat d'instal·lar software addicional. A més, l'aplicació ha d'oferir un tractament d'aquests arxius/dades simple i útil per a l'usuari, amb la informació bàsica de l'exercici realitzat.

Aquest tractament de les dades dels exercicis realitzats ha d'incloure un gràfic on es mostri l'evolució de les dades al llarg de l'exercici realitzat. L'objectiu d'aquesta aplicació no és fer un estudi exhaustiu de les dades sinó mostrar un possible tractament d'aquestes.

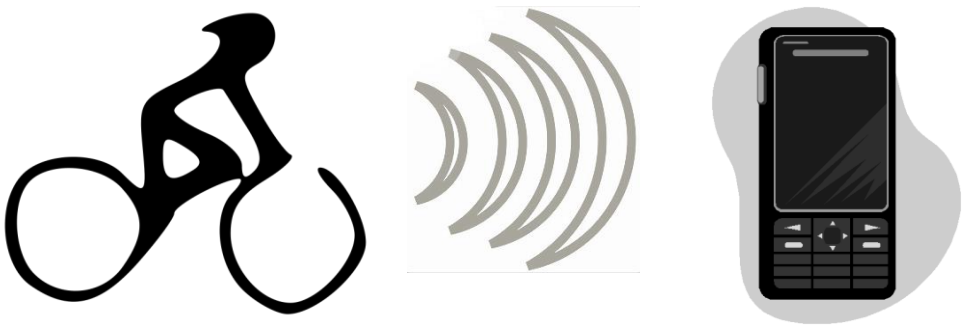

FIGURA 1: esquema general del projecte

1.2 ORGANITZACIÓ DE LA MEMÒRIA

L'estructura de la memòria reflecteix el treball realitzat en les diferents fases del projecte explicades en el punt següent. S'ha prestat especial atenció a una divisió acurada i precisa de les diferents tasques a realitzar durant tot el projecte i ajuntar-les en diverses fases complementàries. Per això durant la memòria s'ha descrit amb cura cada fase, amb els seus objectius, l'entorn de desenvolupament de les tasques i la descripció de les mateixes.

Finalitzant la memòria es recullen diversos annexos amb la intenció d'ajudar al lector a comprendre tant el hardware emprat com el software desenvolupat, amb el manual d'usuari.

1.3 METODOLOGIA EMPRADA

En aquest projecte hi ha una gran varietat de tasques a realitzar i de llenguatges molt diferents amb els que programar, com poden ser assemblador o JAVA. Per això és important separar bé les parts de cada fase per arribar a la consecució amb èxit del projecte.

La divisió de les tasques ha estat la següent:

- **Creació d'un client Bluetooth per al mòbil en JAVA:** el projecte sencer gira entorn d'una comunicació client – servidor on la placa de desenvolupament farà de servidor de dades i el dispositiu mòbil farà de client. Per això en aquesta primera fase es vol tenir un primer contacte amb aquest tipus de comunicació i amb les connexions entre dispositius mitjançant Bluetooth.
- **Comunicació entre el client Bluetooth i el PC mitjançant l'adaptador BT- RS232:** després de crear el client de connexió Bluetooth per al dispositiu mòbil, l'utilitzarem en aquesta fase juntament amb un senzill software escrit en C per al PC, per establir una comunicació sèrie. És un pas previ a realitzar la mateixa connexió entre la placa de desenvolupament i el terminal mòbil, però com es realitza amb el PC, resultarà més senzill i aportarà una valuosa experiència. Per aconseguir comunicar el PC amb el telèfon mòbil s'ha de configurar l'adaptador Bluetooth – RS-232[2], objectiu també d'aquesta fase.
- **Desenvolupament del firmware al simulador PROTEUS:** un cop s'ha configurat l'adaptador Bluetooth i s'ha establert una comunicació sèrie entre el PC i el telèfon mòbil, en aquesta etapa es desenvoluparà tot el firmware que durà el microcontrolador de la placa de desenvolupament. Aquest firmware s'encarregarà de captar les dades generades pels elements de la placa i enviar-les pel canal sèrie. El desenvolupament no es realitzarà directament sobre la placa sinó que s'utilitzarà el software simulador de circuits anomenat PROTEUS. S'ha escollit fer-ho així perquè aquest software és molt realista i, a més, ens permetrà comprovar el funcionament i el valor dels registres durant l'execució del programa. Una altra de les raons principals és el fet de no abusar

de l'ús de la placa de desenvolupament ja que és propietat de la Facultat d'Informàtica de Barcelona.

- **Adaptació del firmware i connexió entre la placa de desenvolupament i l'adaptador BT:** finalitzat el desenvolupament del firmware, en aquesta fase es tracta d'adaptar-lo a la realitat, descarregar-lo al microcontrolador, connectar l'adaptador Bluetooth a la placa de desenvolupament i fer els primers assaigs de comunicació amb el client JAVA del terminal mòbil. En aquest punt del projecte el microcontrolador capta les dades provinents dels sensors emulats, les envia a un dispositiu mòbil que les rep i els aplica un tractament, és a dir, les mostra per pantalla.
- **Desenvolupament de l'aplicació pel dispositiu mòbil:** un cop establerta la comunicació entre la placa de desenvolupament i el dispositiu mòbil, la darrera fase està dedicada en exclusiva a realitzar l'aplicació del terminal mòbil per a un tractament que compleixi els objectius del projecte.

Com es pot veure, les fases que conformen aquest projecte són molt diferents entre elles. Algunes fases són de desenvolupament i d'altres són més tècniques, és a dir, de treball de laboratori.

1.4 JUSTIFICACIÓ

Com ja s'ha esmentat durant la introducció, en aquest projecte es pretén captar i gestionar les dades produïdes pels sensors instal·lats a la bicicleta. Aquests sensors poden avaluar molts paràmetres referents a la bicicleta o a l'usuari i en aquest projecte en concret, l'esforç que realitza l'usuari i les seves pulsacions.

Mesurar aquests valors pot permetre algunes aplicacions, com per exemple aplicacions mecàniques. Pot haver-hi una persona fent rehabilitació amb aquesta bicicleta elèctrica, ja que els motor elèctric la pot ajudar a l'hora de pedalejar. Saber l'esforç físic en temps real possibilita controlar en cada moment la força que genera el motor elèctric i llavors ajudar en major o menor grau a l'usuari durant el seu exercici.

Una altre camp d'aplicació podria ser el mèdic. En aquest cas, saber els valors d'aquests paràmetres pot ajudar a un metge a diagnosticar algun problema al cor del pacient.

Finalment, en el camp comercial es podria aplicar a un entrenador personal que avaluï gràcies a les dades generades, el teu estat de forma i pugui proposar augmentar la duresa de les proves físiques a les que sotmet a l'usuari.

1.4.1 ELECCIÓ DE LA PLACA DE DESENVOLUPAMENT

Abans de començar el projecte la idea era fer servir sensors desenvolupats per altres departaments. Degut al retràs i possibles problemes de temps, es va optar per emular-los. El meu tutor, Joan Climent, va proposar fer servir la placa de desenvolupament que es fa servir a l'assignatura que ell mateix imparteix. Jo ja coneixia la placa de desenvolupament perquè vaig tenir contacte quan vaig cursar l'assignatura. Així que em va semblar bona idea, ja que aquesta placa de desenvolupament conté molts elements connectats a un microcontrolador de gamma mitja suficient com per realitzar aplicacions força complexes.

Entre els elements de la placa de desenvolupament hi ha dos potenciòmetre que es poden utilitzar per fer variacions en la generació de les dades. Un forma part d'un circuit amb dues components analògiques, un sensor de intensitat de la llum i un potenciòmetre. Aquest potenciòmetre genera una tensió variable que es pot ajustar manualment, fet que ens dóna llibertat a l'hora d'emular l'esforç que realitza l'usuari durant un exercici.

Per una altra banda, l'altre potenciòmetre pertany a un circuit generador de freqüències. En funció de la posició del potenciòmetre, a la sortida d'aquest circuit s'aplicaran polsos d'ona quadrada d'una freqüència determinada. Aquest potenciòmetre és igual que el del generador de tensió variable així que també possibilita poder variar la freqüència a plaer. Aquest és un element idoni per emular les pulsacions de l'usuari durant l'exercici.

Un dels objectius és transmetre dades a un altre dispositiu així que també és important remarcar que la placa de desenvolupament disposa de diverses formes de comunicació amb altres dispositius, com per exemple el canal sèrie RS-232.

1.4.2 PER QUÈ ES REALITZA LA COMUNICACIÓ VIA BLUETOOTH?

Avui dia trobem dos grans especificacions sense fils al mercat de les telecomunicacions.

Per una banda tenim el Wi-Fi. El Wi-Fi és un conjunt d'estàndards per a comunicacions locals sense fils (WLAN, Wireless Local Area Network) on el més estès funciona en el rang de freqüència 2,4GHz. Aquest tipus de comunicació té moltes aplicacions tant a l'oficina com a casa o al carrer. Proporciona diverses avantatges com una connexió forta, ràpida i d'ampla cobertura.

Per una altra banda tenim el Bluetooth, que és un altre estàndard de comunicació indicat per a xarxes d'àmbit personal (PAN, Personal Area Network) que funciona també en el rang de freqüència 2,4GHz. Les principals característiques del Bluetooth són el baix preu i consum, i no té res a envejar en quant aplicacions al Wi-Fi, ja que el podem trobar a telèfons mòbils, perifèrics de consoles i ordinadors, impressores i càmeres digitals entre d'altres.

El Wi-Fi respecte al Bluetooth proporciona més seguretat, ja que es pot disposar d'un firewall extern i encriptació avançada com WEP, WPA i WPA2. No obstant, amb software especialitzat s'han aconseguit alts percentatges d'èxit en l'obtenció de les contrasenyes. La transmissió de dades és més ràpida entre 11 i 72 Mbps mentre que el Bluetooth està entre 1 i 3 Mbit/s. El Wi-Fi fa servir més energia, fet que li permet connexions més fortes amb les bases i els dispositius. El radi de funcionament d'una antena Wi-Fi pot arribar teòricament als 300m, però les parets i la disposició dels equips fa que normalment estigui al voltant dels 100m. El radi corresponent al Bluetooth va de 1m per als dispositius de Classe 3 fins als 100m per als dispositius de Classe 1.

Analitzant aquesta petita comparativa entre les dues tecnologies sembla clarament millor el Wi-Fi que no pas el Bluetooth. No obstant, parlant sobre estàndards de telecomunicació i en general, hauríem d'evitar els termes millor/pitjor i utilitzar apropiat/no apropiat. Encara que no ho sembli, el Bluetooth és més escaient per al nostre projecte. A continuació es justifica el per què.

- Com s'ha esmentat en fases anteriors, en aquest projecte el consum d'energia és molt important per que la bicicleta ha de funcionar amb bateries que han de subministrar energia el màxim temps possible. Per això un dels principals avantatges del Bluetooth és el seu baix consum. Un consum d'energia baix implica un senyal més dèbil, però no suposa un greu problema ja que les distàncies de funcionament són molt curtes en teoria.

- Tot i que la seguretat del Bluetooth no sigui alta, les dades amb les que treballarem no són de vital importància. No treballarem amb números secrets del nostre compte corrent ni manipularem informació privada. A més, aquesta aplicació es suposa que no serà atacada per cap delinqüent informàtic, ja que no pot treure profit de les dades que pot interceptar. La seguretat no és de gran importància.

- La velocitat de transmissió del Bluetooth és suficient amb molt marge per al nostre projecte, ja que només utilitzem de l'ordre de 0,02Kb cada segon. Els sensors capten dades que enviem al dispositiu mòbil, volem les pulsacions o l'esforç físic de l'usuari però no cada mil·lèsima de segon, si no cada cert temps prudencial. Llavors, és innecessari un gran ample de banda com el del Wi-Fi.

- En teoria, aquest projecte està dissenyat per què l'usuari porti ell mateix el dispositiu mòbil mentre està realitzant l'exercici amb la bicicleta. Llavors, el radi de funcionament de la comunicació hauria de ser d'un metre aproximadament. Això es correspon amb un dispositiu Bluetooth de Classe 3. No ens calen centenars de metres de cobertura.

- M’agradaria afegir altres raons de pes per a l’elecció del Bluetooth com a estàndard de comunicació, com per exemple l’àmplia distribució de dispositius mòbils que disposen de Bluetooth en els darrers anys. El Wi-Fi tot just està començant en aquest camp.
- Per una altra banda tenim que el funcionament del Bluetooth és molt senzill i facilita les comunicacions entre dispositius mòbils.
- Finalment, i des del punt de vista econòmic del projecte, una altra avantatge molt important és el seu baix cost que fa que la fabricació de la bicicleta sigui més assequible mantenint les característiques de la connexió sense fils.

Per aquests motius, el Bluetooth és la tecnologia idònia per dur a terme aquest projecte, i només ens cal l’adaptador sèrie RS232 – Bluetooth a la placa de desenvolupament per poder comunicar la bicicleta amb el dispositiu mòbil de l’usuari.

1.4.3 APROFUNDINT EN L’ESTÀNDARD BLUETOOTH

El Bluetooth és una especificació industrial per les Xarxes d'Àmbit Personal (PAN, Personal Area Network) sense fil, bàsicament és refereix a que serveix per connectar els dispositius que podem portar a sobre o a una distancia pròxima.

Amb el Bluetooth, podem obtenir una forma de connectar i intercanviar informació entre dispositius com ordinadors de butxaca, telèfons mòbils, ordinadors portàtils, ordinadors, impressores i càmeres digitals a través d'una forma segura, de baix cost a través d'ones de ràdio de baixa freqüència.

El Bluetooth permet a aquests dispositius comunicar-se quan estan a l'abast, encara que no estiguin a la mateixa habitació, fins a un límit de 100 metres entre ells depenent de la classe de potència del producte. Els productes poden estar disponibles en alguna d'aquestes tres classes de potència:

- Classe 3 (1 mW) és la més rara, i ens permet una transmissió de 10 centímetres a un màxim de 1 metre.
- Classe 2 (2.5 mW) és més comuna i ens permet una distància de fins a 10 metres.
- Classe 1 (100 mW) té l'abast més gran, de fins a 100 metres tot i que el consum més elevat.

Utilitats

Des del punt de vista dels ordinadors de butxaca, el Buetooth és útil per imprimir, connectar a Internet i enviar missatges, fotos o fax amb el telèfon mòbil o un altre ordinador. En cada cas

caldrà disposar d'una impressora, telèfon mòbil o ordinador que disposin d'una connexió Bluetooth.

També permet connectar l'ordinador de butxaca a uns auriculars, un lector de MP3, una càmera de fotos o un GPS, per exemple. Un dels avantatges per a un usuari d'ordinadors de butxaca, per exemple, és poder fer servir els mateixos auriculars per al telèfon mòbil que per escoltar MP3.

Des de la seva creació s'ha intentat promoure que els dispositius tècnics que permeten incloure aquesta connexió tinguessin un preu molt assequible, baix consum i fossin aparells petits. El preu actual d'un xip que permet incloure aquesta connexió ronda els 5 Euros. A mesura que és més fàcil aconseguir-los, el preu dels xips ha anat baixant.

Això fa que cada cop més, els telèfons d'alta i mitja gamma incloguin de sèrie aquesta connexió sense fil. Si els aparells portàtils segueixen disminuint de mida, i les necessitats de connexió entre ells augmenta, aquesta tecnologia té més probabilitats de consolidar-se.

Cada cop més ordinadors de butxaca, telèfons mòbils, impressores i ordinadors disposen de connexió Bluetooth.

Història

El Bluetooth va ser impulsat per un grup d'empreses, encapçalat per Nokia, Ericsson, IBM, Intel i Toshiba, anomenat Grup d'Interès de Bluetooth (SIG) l'any 1999. Més tard centenars d'empreses s'hi han afegit (com ara One2One, Motorola, Qualcomm, Compaq, Dell, Intel, Microsoft, 3Com Palm, VLSI, Xircom, Psion Dacom i Lucent). Va agafar el seu nom del rei danés Viking Harald Blåtand (Bluetooth en anglès).

Tecnologia

El Bluetooth treballa en una banda lliure de l'espectre electromagnètic, per a la qual no cal llicència (a 2.45 GHz). La velocitat de transmissió teòrica és de 1 Mbps (versió 1.x) tot i que la màxima en un sol sentit amb una connexió asíncrona és de 720 Kbps. El Bluetooth porta incorporades diverses mesures de seguretat.

A més de les versions prototip, hi ha hagut tres versions aprovades del protocol, la versió 1.0, amb molts errors de interoperabilitat; la versió 1.1, que va solucionar els principals problemes de la versió anterior; i la versió 1.2, més orientada a multimèdia i que millora la fiabilitat i la seguretat. Cada nova versió és compatible amb les anteriors, és a dir, permet connectar amb tots els dispositius de les versions anteriors, en les condicions que aquests necessitin. La nova versió 2.0 es va publicar a principis del 2006 i millora la velocitat de

transmissió fins a 3 Mbps aplicant EDR (Enhanced Data Rate). El nom adoptat pel IEEE[4] de l'estàndard és 802.15.1.

Es tracta d'un estàndard obert, tot i que els seus drets pertanyen al grup SIG i per tant ningú que no hi pertanyi pot construir un aparell amb la seva funcionalitat.

Consum

Un dels avantatges dels xips amb tecnologia Bluetooth és el seu baix consum. Per exemple, un xip Wi-Fi consumeix uns 7 watts, i un de Bluetooth classe 2 (per exemple un mòbil) cap a 2.5 mW, menys de una mil·lèsima part.

Implantació

En l'actualitat gairebé tots els ordinadors de butxaca, ordinadors portàtils i telèfons mòbils nous són compatibles amb tecnologia Bluetooth de sèrie.

Molts fabricants de cotxes inclouen també com a opció equipament de mans lliures basat en Bluetooth, sovint acompanyat de control per veu, ja que és una opció més còmoda que els mans lliures tradicionals on no cal fer cap connexió ni prémer cap botó per començar a fer servir el mans lliures.

CAPÍTOL 2

CREACIÓ DEL CLIENT PER AL MÒBIL EN JAVA

2.1 OBJECTIU

Fent un anàlisi del que requerirà el projecte, en la primera fase ens centrarem en l'arquitectura i la comunicació. L'arquitectura serà del tipus client – servidor, on la placa de desenvolupament farà de servidor i el dispositiu mòbil farà de client. Per aquesta raó a la primera fase l'objectiu és familiaritzar-se amb JAVA Microedition[5] i més en concret amb l'API[6] del Wireless Toolkit 2.5.2 que inclou la llibreria JSR 82 (J2ME i WTK). A més a més, també s'aprofundirà en la comunicació client-servidor i la connexió via Bluetooth entre dispositius.

Així durant aquesta fase es desenvoluparà una primera aplicació per realitzar la part de comunicació via Bluetooth del dispositiu mòbil. Aquest codi desenvolupat serà la base sobre la qual es realitzarà l'aplicació final del telèfon mòbil. Durant aquesta etapa del projecte no es farà servir encara la placa de desenvolupament i el seu lloc l'ocuparà un altre dispositiu mòbil. Un cop realitzada amb èxit aquesta fase, es procedirà més endavant a reemplaçar la funció d'aquest dispositiu mòbil per la placa real.

2.2 ENTORN

Per desenvolupar aquesta aplicació s'ha fet servir un editor de text avançat especialitzat per a la programació i la línia de comandes de Windows per integrar al projecte les llibreries J2ME amb el WTK i el meu codi font. A més he aprofitat l'emulador del WTK. Aquest emulador permet emular un dispositiu mòbil a l'ordinador sense haver de fer el traspàs de l'aplicació de l'ordinador al dispositiu mòbil. Conseqüentment, és més pràctic fer proves i avaluar el funcionament de l'aplicació.

Un cop avaluat el correcte funcionament de la aplicació amb l'emulador JAVA, es transmet als dispositius mòbils reals per continuar les proves. El dispositius que s'han fet servir i que també formen part de l'entorn d'aquesta fase són un Sony Ericsson K800i, un Sony

Ericsson W880i i un ordinador portàtil. Cal remarcar que encara que els dos models de telèfon siguin de la mateixa marca és pura coincidència i no afecta al funcionament de l'aplicació.

2.3 DESCRIPCIÓ

Com he esmentat abans, el projecte seguirà l'estructura client-servidor. En aquesta primera fase, on encara no es farà servir la placa de desenvolupament, el seu rol servidor el farà un altre dispositiu mòbil. Així es desenvoluparà una aplicació per als dispositius mòbils on un dispositiu farà de servidor i un altre farà de client. Un cop establerta la connexió, el servidor enviarà certa informació al client, que s'encarregarà de tractar-la.

2.3.1 ESTRUCTURA DE L'APLICACIÓ

Primer de tot, tractem l'estructura que seguirà l'aplicació. Aquesta estructura es basa en una relació entre dues entitats: el servidor, que ofereix un recurs de qualsevol tipus (físic, de dades, etc) a l'altre (el client) per què aquesta en tregui un profit o avantatge.

Característiques d'un servidor:

- Passiu (esclau)
- Espera peticions
- Al rebre una petició la processa i retorna una resposta

Característiques d'un client:

- Actiu (mestre)
- Envia peticions
- Al enviar una petició espera una resposta i quan la rep, la processa

Com s'ha vist, el funcionament de l'aplicació serà de tipus client-servidor, i el protocol de comunicació entre les dues bandes serà Bluetooth, tal i com es descriu prèviament i degut a les necessitats del projecte. La llibreria WTK ens proporciona les classes per a desenvolupar aquesta comunicació. La connexió i posterior comunicació Bluetooth en el nostre cas seguirà aquest esquema:

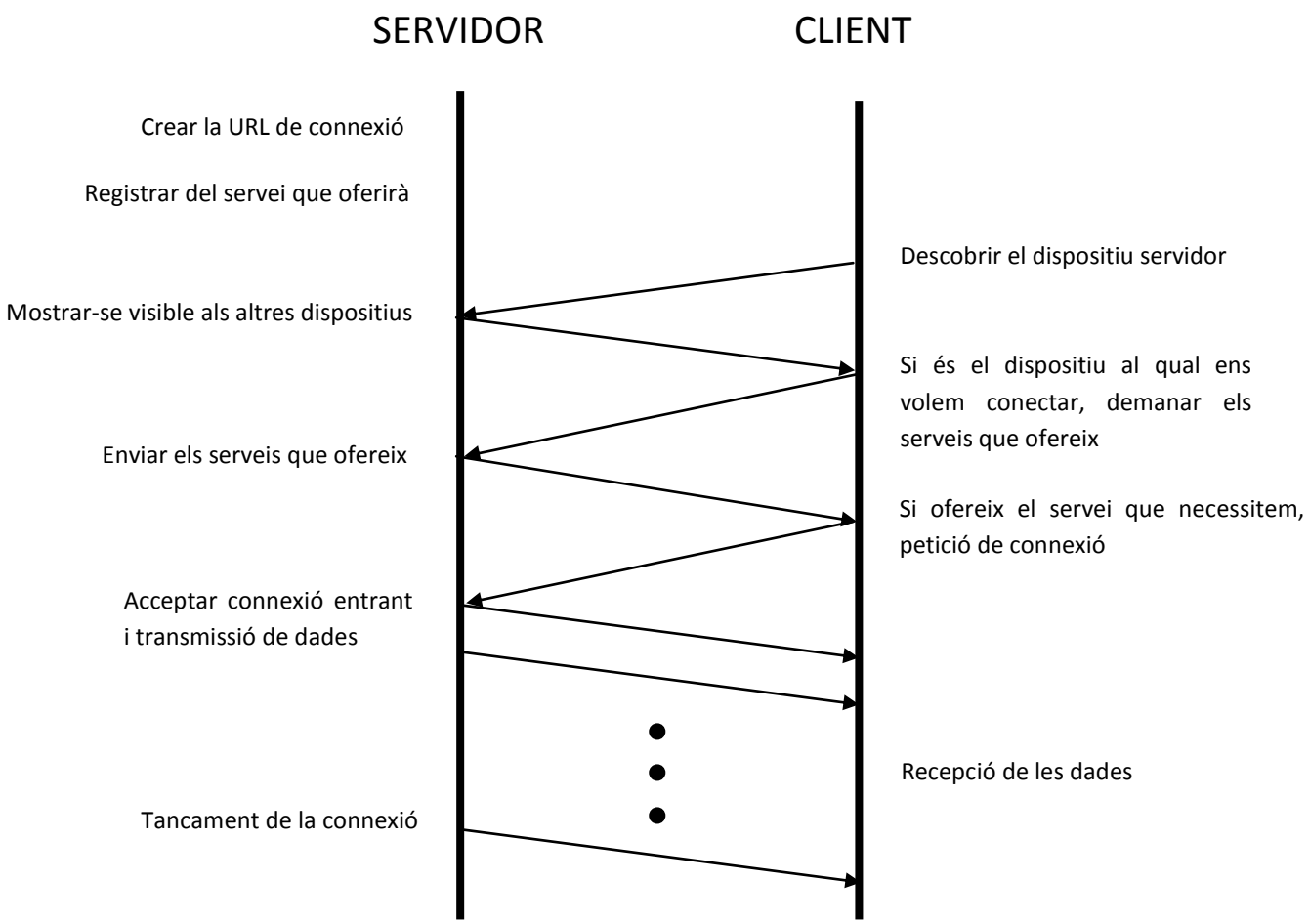

FIGURA 2: Comunicació client - servidor de l'aplicació

Part servidor

1. Creació de la URL[7] de connexió.

Primerament, en el servidor hem d'establir la URL a la qual es connectaran els clients. Com a protocol especifiquem "btssp" (SSP, Serial Service Profile), que és el protocol per emular la transmissió tipus sèrie de dades a través de radiofreqüència. El prototip d'URL que es genera és així:

```
btspp://localhost:" + RFCOMM_ID + ";name=PROJECT
```

2. Registre del servei del port sèrie.

Ara, l'aplicació inicialitza el servei que oferirà cridant a la funció `connector.open(URL)` on "URL" és l'String amb l'adreça URL creada al pas anterior. A partir d'ara, la resta de dispositius poden accedir a aquest servei.

3. Acceptar connexions i enviament de dades.

Tal i com farà la placa de desenvolupament, acceptarem connexions entrants i començarem a enviar les dades captades pels sensors, com els batecs i l'esforç. Per això obrim un canal de comunicació de sortida per on enviem les dades que volem que rebi el client. És molt semblant a com es fan servir els canals de sortida d'un sistema operatiu, per exemple.

4. Tancament de la connexió.

Per finalitzar l'operació per part del servidor tanquem la connexió invocant el mètode `close()`.

Part client

1. Descobrir dispositius.

Qualsevol aplicació, en el nostre cas el client, pot obtenir una llista dels dispositius al seu abast fent servir la funció `startInquiry()`, aquesta funció retornarà una llista dels dispositius que trobi al seu abast.

2. Descobrir serveis.

En el següent pas el client descobreix els serveis disponibles per a cada dispositiu trobat. Com es fa servir el mètode `startInquiry()` aquest pas es pot fer simultàniament amb el primer, ja que la funció de cerca no és bloquejant.

3. Petició de connexió.

Un cop trobat el dispositiu al qual ens volem connectar i que ofereix el servei que necessitem, establim la connexió a partir de l'adreça URL del servei al servidor.

4. Transferència de dades

Quan ja tenim la connexió establerta, ja només cal rebre les dades. El servidor ofereix un canal de comunicació en dos sentits, per on s'enviaran i es rebran les dades. Com he dit abans el seu funcionament és semblant a una *pipe* de sistema operatiu que permet llegir pel canal d'entrada amb el mètode `read()`. Un cop s'obtinguin les dades, aquesta aplicació les tractarà mostrant-les per pantalla.

2.3.2 CREACIÓ DE L'APLICACIÓ

Un cop explicada l'estructura del programa i el seu funcionament bàsic, a continuació s'aprofundirà en el desenvolupament d'una aplicació per a un dispositiu mòbil basat en J2ME. Per crear aquest MIDlet[8] ens hem basat en el cicle llarg, que consta de les següents passes:

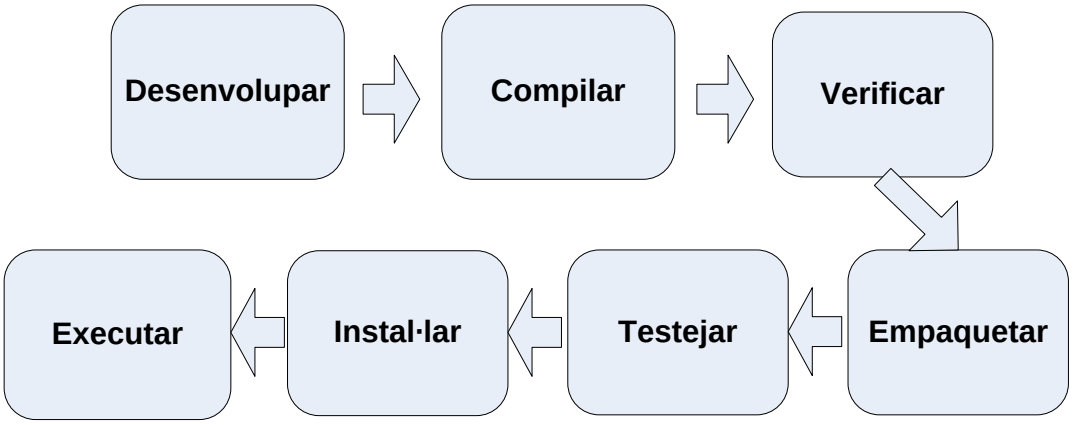

FIGURA 3: Fases en la creació de MIDlets

Disseny del codi font

Crear MIDLets és força diferent del que he programat fins al moment, bàsicament perquè el programa s'executa en un entorn diferent, un dispositiu mòbil que disposa de menys recursos que un ordinador convencional. El llenguatge de programació és J2ME, que és JAVA amb menys opcions pel tema dels recursos. Això es nota a l'hora d'utilitzar algunes classes, que veuen retallades les seves funcions vers al JAVA.

Els MIDlets tenen un cicle de vida predeterminat i estàndard amb diversos estats. Fet que normalment no succeeix en una aplicació JAVA per a PC.

A més, els programes convencionals que havia fet fins ara consistien en una funció principal a partir de la qual es realitzaven totes les operacions, potser cridant altres classes. Els MIDlets són diferents, es basen més en pantalles i les seves accions que no pas en una funció principal. Per això una part molt important del desenvolupament de l'aplicació són les pantalles interactives amb l'usuari.

A partir d'aquestes pantalles l'usuari escollirà el rol que vulgui, client o servidor, i rebrà els resultats a la pantalla del dispositiu. El codi, encara que sigui una aplicació senzilla, és massa extens i complex com per posar-ho en aquesta part de la memòria, tot i que a continuació explicaré per sobre alguns detalls del codi. Per veure el codi complet, es pot consultar el cd adjunt.

Un MIDLet sobrecarrega 3 funcions bàsiques: `startApp()`, `pauseApp()`, i `destroyApp(boolean unconditional)`. Això per la part de MIDlet, per la part de la connexió Bluetooth necessitem implementar els *listeners*[9] esmentats amb anterioritat.

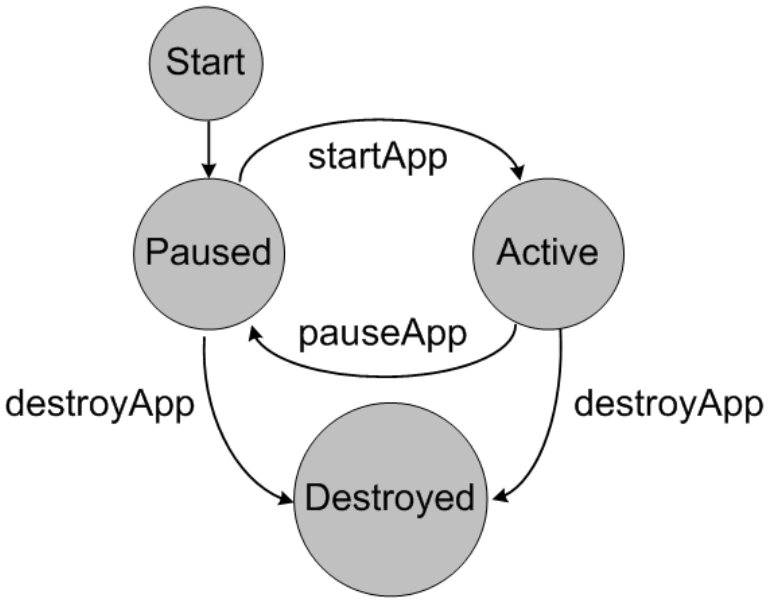

FIGURA 3: Cicle de vida d'un midlet

Amb aquesta aplicació es vol aconseguir comunicar dos terminals mòbils mitjançant Bluetooth. Un dispositiu farà de servidor i l'altre farà de client seguint l'estructura vista amb anterioritat. Així, un cop establerta la connexió entre els dos dispositius, el servidor enviarà dades al client, tal i com es pretén fer en aquest projecte, on la bicicleta (servidor) enviarà dades al telèfon mòbil (client).

Més endavant es precisarà el funcionament de cara a l'usuari amb exemples gràfics de les pantalles, però es pot avançar que el programa parteix d'un menú inicial on s'escull el rol que es vol seguir durant la comunicació i llavors enviarà o rebrà dades i les mostrarà per pantalla.

Llavors, un cop desenvolupat el codi font a l'editor de text, el guardem com a arxiu java, per exemple amb el nom BlueToothExp.java.

Compilació del codi font

Un cop tenim l'arxiu .java amb el codi, l'hem de compilar per fer-lo servir en dispositius mòbils. Compilar un MIDLet no és gaire diferent a compilar un programa en JAVA. Executem el mateix compilador javac. La diferència resideix en el CLASSPATH[10], ja que ara no podem fer servir les classes "normals" que fa servir JAVA, sinó que hem de fer servir les del J2ME. Les més

importants i que es necessiten són la CLDC[11] and MIDP[12] classes. Llavors, un cop hem fet que el CLASSPATH apunti al directori on estan aquestes classes, compilem mitjançant la línia de comandes de la manera següent:

```
$>javac -target 1.4 -source 1.4 com/j2me/part1/BlueToothExp.java
```

Verificació

La nostra nova classe ha de ser verificada. La verificació del codi executable és un pas que fa la JVM (Java Virtual Machine) previ al funcionament per assegurar-se de que l'arxiu .class és correcte tan estructural com conceptualment per a la especificació de la màquina virtual. Per fer aquest pas també utilitzem la línia de comandes amb la següent comanda:

```
$>preverify com.j2me.part1.BlueToothExp
```

Així es crearà per defecte un arxiu .class amb la nostra classe verificada a dins d'una carpeta anomenada output. La verificació preserva l'estructura de paquets si la nostra classe en conté. Es pot canviar la ruta per defecte amb l'opció –d.

Empaquetament

Hi ha moltes passes per empaquetar el MIDLet i s'han de seguir en ordre. El primer pas és crear l'arxiu "Manifest". Aquest arxiu de text descriu el contingut del Java Archive (JAR) que crearem a continuació. Es poden declarar molts atributs en aquest arxiu, però per la nostra aplicació ja tindrem suficient amb aquests:

MIDlet-Name:	BlueToothExp
MIDlet-Version:	1.0.0
MIDlet-Vendor:	Jose Carlos Yáñez Nieto

Guardem aquest arxiu a la carpeta "output" que s'ha creat en el pas anterior.

A continuació creem l'arxiu .jar que empaqueta la classe verificada i el Manifest. Des de la carpeta "output" executem la comanda següent que ens crearà un arxiu .jar a la carpeta en la que estem.

```
$>jar cvfm BlueToothExp.jar manifest.mf .\com
```

El següent pas és crear l'arxiu Java Aplication Descriptor (JAD). Aquest arxiu descriu com el J2ME del dispositiu ha d'instal·lar l'aplicació. Un altre cop, aquest arxiu de text conté diversos atributs. Per la nostra aplicació especificarem aquests:

MIDlet-1:	BlueToothExp, , com.j2me.part1.BlueToothExp
MIDlet-Description:	MIDlet client/servidor
MIDlet-Jar-URL:	BlueToothExp.jar
MIDlet-Jar-Size:	6515
MIDlet-Name:	BlueToothExp
MIDlet-Permissions:	javax.microedition.io.Connector.bluetooth.client, javax.microedition.io.Connector.bluetooth.server
MIDlet-Vendor:	Jose C. Yáñez Nieto
MIDlet-Version:	1.0
MicroEdition-Configuration:	CLDC-1.0
MicroEdition-Profile:	MIDP-2.0

Guardem aquest arxiu a la mateixa carpeta. És important l'atribut MIDLet-Jar-Size. El valor d'aquest camp ha de ser exacte i coincidir amb la mida de l'arxiu .jar.

Amb aquest pas completem l'empaquetament. Ara podem testejar la nostra aplicació.

Testejar

Després de tots aquests passos, podem executar la nostra aplicació i provar-la. El WTK de JAVA incorpora un emulador de dispositius mòbils. Ara, si executem l'arxiu .jad arrencarà automàticament l'emulador com es mostra en aquesta imatge. Com he explicat abans, aquest emulador simplifica el fet de fer canvis al codi font i provar-los en el dispositiu, ja que no cal

enviar l'executable al dispositiu real, instal·lar-lo i fer proves. A més, l'emulador disposa de consola amb la qual podem *debuggar* l'aplicació.

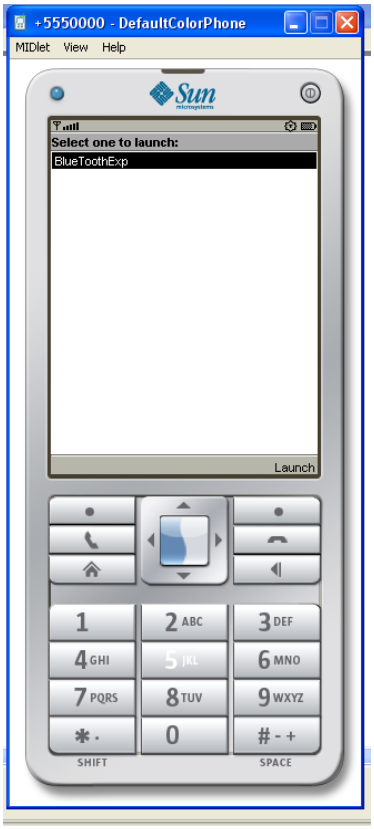

FIGURA 4: Emulador WTK

Instal·lació de l'aplicació al dispositiu mòbil

Un cop s'han fet proves amb l'emulador instal·lem l'aplicació als dos telèfons mòbils. Per instal·lar-ho només s'ha de transferir el fitxer .jar al dispositiu a través de Bluetooth o del cable USB. Al finalitzar l'enviament del fitxer, el telèfon mòbil ens donarà l'oportunitat d'instal·lar el software. Només s'han de seguir les instruccions que apareixen a la pantalla.

FIGURA 5: Pantalla de l'aplicació

Execució

Un cop tenim el software als dispositius amb els que es vol treballar, només hem d'executar l'aplicació al nostre telèfon mòbil. Com s'ha explicat abans, hi ha dos rols a repartir en aquesta aplicació: client o servidor. A la primera pantalla de l'aplicació podem escollir quin rol volem jugar. Si escollim el rol servidor surt un missatge a la pantalla del nostre

dispositiu demanant l'autorització per establir una connexió de servidor. Això és una mesura de seguretat pròpia del dispositiu mòbil per assegurar-se de que no s'estableix un servidor sense el consentiment de l'usuari. Si acceptem en el missatge, s'arrenca el servidor i ja espera peticions de connexió. Per informar l'usuari d'aquest fet mostrem un text a la pantalla del dispositiu tal i com es pot veure a la imatge de més avall. El servidor acceptarà totes les peticions de connexió que rebi i els hi enviarà un text curt primer, s'esperarà uns segons i enviarà un altre text.

Ara que ja tenim el servidor esperant a rebre peticions de connexió, al menú inicial de l'altre dispositiu executem el client que buscarà serveis SPP de dispositius Bluetooth propers. En aquest cas, com es pot veure a la imatge, troba el dispositiu anomenat "WirelessToolkit". Si ens connectem a aquest servei (que és el servidor) a la pantalla del mòbil ens sortirà l'avís de que s'establirà la connexió client-servidor. Igual que abans, els missatges de confirmació són una mesura de seguretat implícita en els telèfons mòbils.

FIGURA 6: Pantalla on es mostren els dispositius trobats

A la banda del servidor hem d'acceptar un altre cop el missatge de confirmació per establir la connexió. Llavors, si l'autoritzem, quan es connecti el client ens apareixerà un missatge d'alerta que informa a l'usuari de la connexió d'un nou client al servidor.

Finalment, quan el client es connecti rebrà automàticament el text que li envia el servidor i el mostrarà per pantalla, i al cap d'uns segons, mostrarà el següent missatge enviat pel servidor tal i com mostra la imatge següent:

FIGURA 7: Dades rebudes

2.4 CONCLUSIÓ

En aquesta fase es té un primer contacte amb el món de les aplicacions per a terminals mòbils i les comunicacions Bluetooth entre ells. L'experiència adquirida durant aquesta primera fase serà molt útil per al futur desenvolupar l'aplicació final. Es creen les bases del que serà la comunicació per part del dispositiu mòbil amb la placa de desenvolupament.

En aquest moment estem en un punt en que la comunicació es realitza entre dos dispositius mòbils. Llavors, a partir d'aquí, i durant les properes fases del projecte, es tractarà de substituir el dispositiu que té el rol servidor per la placa de desenvolupament.

CAPÍTOL 3

COMUNICACIÓ ENTRE EL CLIENT BLUETOOTH I EL PC MITJANÇANT L'ADAPTADOR BT- RS232

3.1 OBJECTIU

Un cop finalitzada la primera fase amb la creació d'un client senzill Bluetooth pel dispositiu mòbil que es connecta a un servidor i mostra les dades que rep, comencem la segona fase. L'objectiu d'aquesta segona fase és establir una comunicació Bluetooth entre l'ordinador i el mòbil a través de l'antena que farem servir amb la placa de desenvolupament. Un cop feta la comunicació, caldrà crear un programa senzill en C que generi dades i les enviï pel port sèrie a l'adaptador, que automàticament les enviarà per Bluetooth telèfon al mòbil ja connectat. L'aplicació del dispositiu mòbil canviarà només en la part del tractament de les dades, ja que es connecta igual al dispositiu Bluetooth. Aquesta fase ens permetrà conèixer en profunditat el funcionament de l'adaptador port sèrie cap a Bluetooth que es farà servir amb la placa de desenvolupament. A més, es realitzaran els primers enviaments i tractaments de paquets de dades de forma constant, en un *stream*[13], tal i com ho farà la placa de desenvolupament a partir de les dades captades pels sensors.

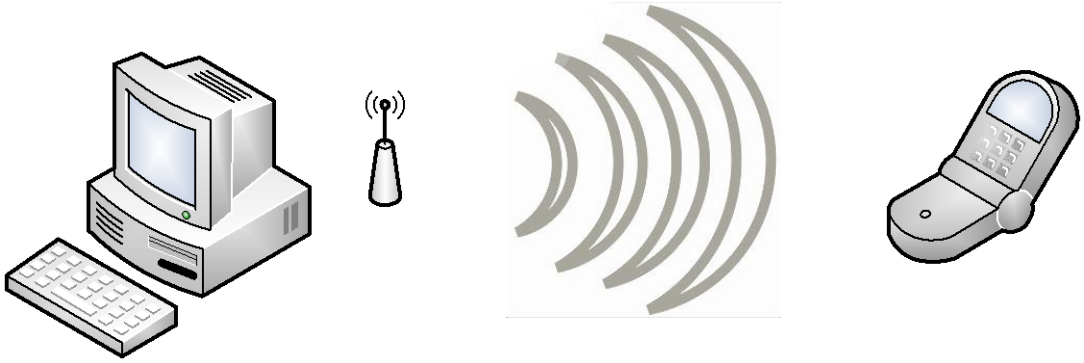

FIGURA 8: Comunicació a la Fase 2

3.2 ENTORN

El nou hardware que s'ha fet servir durant aquesta fase és un adaptador Bluetooth – RS232. Aquest perifèric rep les dades per un port sèrie i les envia a través de Bluetooth al dispositiu que estigui connectat. Per a una descripció més acurada d'aquest dispositiu, es pot consultar l'annex D.

Per dur a terme les tasques d'aquesta segona fase, com per exemple la configuració de l'adaptador Bluetooth, s'ha fet servir el software de configuració anomenat ParaniWIN proveït pel fabricant i el Hyper Terminal de Windows. A més, per implementar el programa que genera i envia dades pel port sèrie del PC, he fet servir l'editor Dev-C++ que ja incorpora un compilador de C. Com a la primera fase, s'utilitzarà un editor de text avançat per fer les modificacions a l'aplicació del dispositiu mòbil. Finalment, mitjançant la línia de comandes de Windows s'executarà aquesta aplicació.

3.3 DESCRIPCIÓ

Per a la consecució d'aquesta fase, primer s'ha de configurar l'adaptador Bluetooth. Tal i com s'indica amb anterioritat, s'ha fet servir el programari que proveeix el fabricant del producte i l'Hyper Terminal de Windows XP. També es pot fer una configuració més bàsica via hardware, però les opcions que ens proporciona són insuficients, com es veurà més endavant.

3.3.1 CONFIGURACIÓ DE L'ADAPTADOR BLUETOOTH – RS232.

La configuració per hardware disposa de 3 interruptors que, combinant-los, ajusten el Baud-rate[14] de 2400 a 115.2K amb diversos nivells intermedis. També permet ajustar l'adaptador perquè la configuració sigui a través de software tal i com mostra la següent imatge:

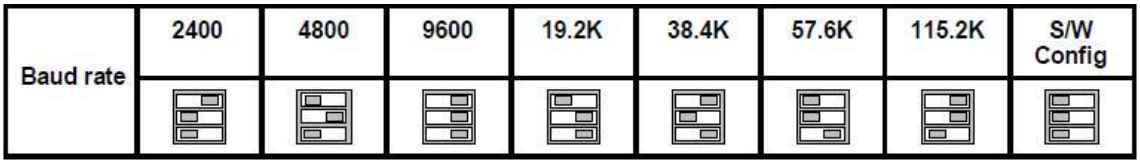

FIGURA 9: Configuracions Hardware al PARANI-SD 200

L'opció escollida és la de configuració per software, ja que ens permetrà ajustar més paràmetres. A més, es disposa d'un interruptor més per l'opció de control de flux per hardware. Aquest control via hardware evita la pèrdua d'informació com a conseqüència de condicions dolentes de transmissió. L'adaptador disposa d'un *buffer* intern d'emmagatzematge de dades

provinents del *host* on està connectat. Llavors envia les dades repetidament fins que tots els paquets s'han enviat. Però quan les condicions no són bones, la transmissió de dades pot ser més lenta i això pot produir que tingui el *buffer* ple, i continuï rebent dades del *host* provocant que l'adaptador funcioni malament.

La solució que aporta el control de flux per hardware, és parar la recepció de dades fins que hi hagi més espai per a dades noves. En aquest projecte prima la latència de dades, és a dir, un paquet en sí no és molt important i es pot perdre sense que afecti a la integritat, ja que els sensors van captant i generant dades contínuament. Així, no serà necessari el control de flux per hardware i s'inhabilitarà.

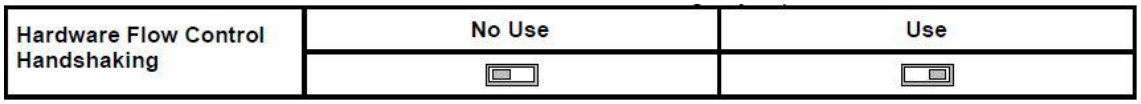

FIGURA 10: Configuració del control de flux

Mitjançant comandes AT es poden modificar els registres 46 registres "S" de la memòria FLASH de l'aparell que controlen molts paràmetres.

Per exemple el registre S3: "UART Policy"

Si S3=0 llavors prima el valor de les dades, que no es perdin. El dispositiu rep dades i les guarda. Llavors les envia en un paquet de dades.

Si S3=1 llavors prima la latència en l'enviament de les dades.

Per editar el valor d'aquest registre mitjançant comandes AT es pot fer servir Hyper Terminal de Windows amb la següent comanda:

```
>ATS03=1
```

Per mostrar el seu valor concret, es pot fer servir aquesta altra comanda:

```
>ATS03?
```

Mitjançant el software ParaniWIN es poden ajustar també molts paràmetres. Es poden establir els paràmetres de la connexió Bluetooth, com el nom del dispositiu, el PIN, i si volem autenticació i encriptació; i també el mode d'operació del dispositiu. Més endavant explicarem en detall aquest paràmetre.

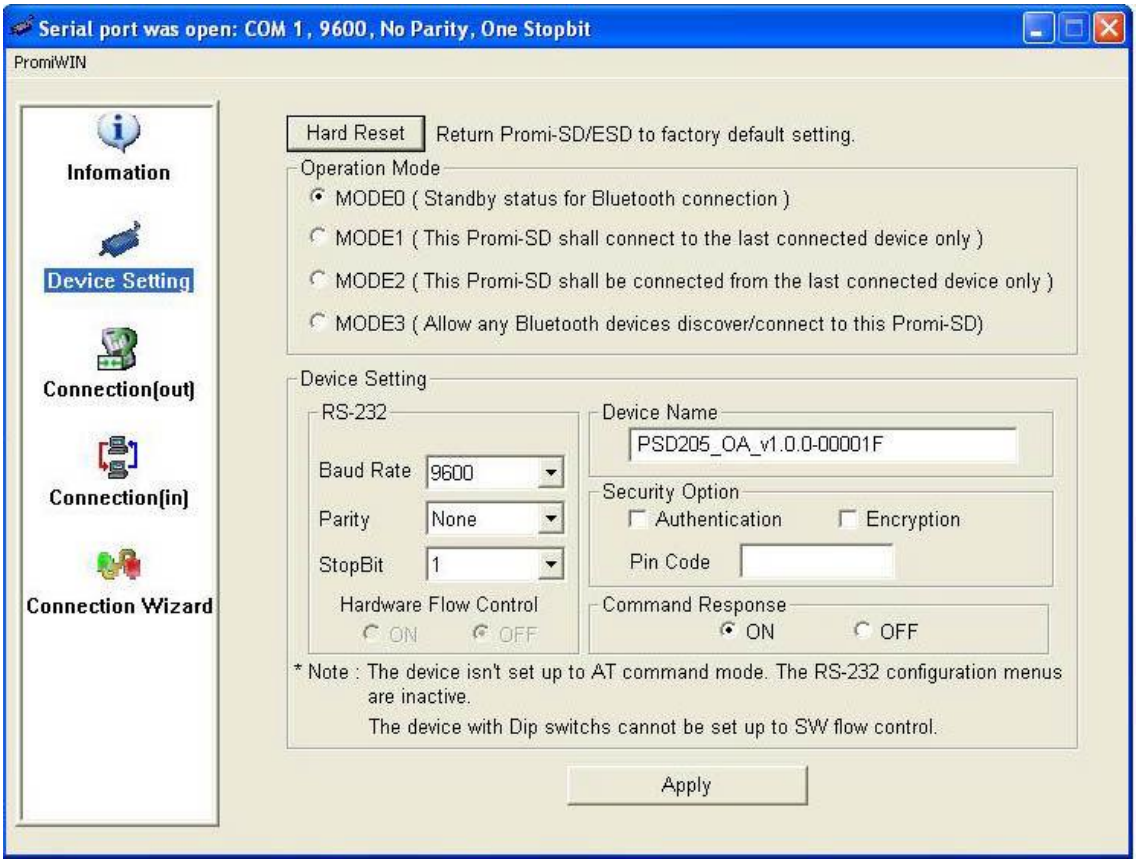

FIGURA 11: Pantalla de configuració PARANIWIN

Primerament, per configurar les opcions del port sèrie, s'ajustarà el Baud-Rate. Aquest valor significa el nombre de senyals per segon de comunicació al port sèrie. En aquest projecte el nombre de dades per segon no és gaire elevat, ja que només s'enviaran paquets cada cert temps amb el valor captat pels sensors de la bicicleta. Així que s'ha optat per un valor intermedi que proporciona una velocitat més elevada de la que necessitem. Farem servir 8 bits (7 bits per caràcter més 1 bit StopBit) i sense paritat.

Com a opcions de seguretat es marcaran les caselles d'autenticació, afegint un codi PIN per al moment d'establir la connexió amb un altre dispositiu, i també l'encriptació de les dades. La motivació per afegir un codi PIN és la possibilitat de limitar l'accés a l'antena Bluetooth per part de persones alienes al projecte. Per exemple, si s'implementés aquest projecte en una bicicleta comercial, es podria facilitar el número PIN a una targeta protegida i així només s'hi podria connectar l'usuari que ha comprat la bicicleta. És un funcionament semblant a un telèfon mòbil amb el seu número PIN.

Per part del mode d'operació del dispositiu tenim diverses opcions:

- Mode 0. En aquest mode l'usuari pot canviar els paràmetres de configuració del dispositiu i és directament controlat per comandes AT. Aquest és el mode per defecte.

- Mode1. L'adaptador, quan està en aquest mode, intenta connectar-s'hi a l'últim dispositiu Bluetooth amb el qual s'ha connectat ja que sempre guarda a una base de dades la seva adreça. En aquest mode, el dispositiu no pot ser descobert o connectat per altres dispositius Bluetooth. Si no hi ha cap adreça a la base de dades no funciona correctament en aquest mode.

- Mode 2. Aquest mode és semblant al mode 1. Al mode 2, el dispositiu espera per una connexió provinent de l'últim dispositiu Bluetooth guardat a la base de dades. Si no hi ha cap registre a la base de dades no funcionarà correctament en aquest mode. No pot ser descobert ni connectat per cap altra dispositiu que no sigui el darrer al qual es va connectar.

- Mode 3. Al mode 3 està a l'espera per a la connexió de qualsevol altre dispositiu Bluetooth. En aquest mode pot ser descobert i connectat per qualsevol dispositiu.

El mode d'operació establert ha sigut el 3 ja que en aquest mode el nostre dispositiu, que està visible i pot rebre connexions, està esperant una connexió d'algun altre dispositiu Bluetooth. En aquest projecte, el mòbil de l'usuari detectarà l'adaptador Bluetooth port sèrie RS232, intentarà connectar-s'hi i haurà d'introduir el codi PIN determinat per l'adaptador. L'usuari de la bicicleta pot canviar de telèfon mòbil o fins i tot un altre membre de la mateixa família podria utilitzar la bicicleta, així que els modes 1 i 2 d'operació no interessen, ja que es connecten sempre a l'últim dispositiu. Evidentment el mode 0 tampoc ens interessa ja que no treballarem amb comandes AT.

Un cop fets els ajustaments comentats anteriorment al dispositiu, es fa una prova de connexió a l'adaptador Bluetooth des d'un telèfon mòbil i la connexió es realitza amb èxit. El següent pas en aquesta fase, un cop configurat l'adaptador Bluetooth sèrie RS232, és la creació d'un senzill programa en codi C amb el que puguem generar dades i enviar-les a traves del port sèrie del PC cap a l'adaptador, i que de l'adaptador vagin automàticament al dispositiu que estigui connectat.

3.3.2 PROGRAMA EN C.

Per desenvolupar aquesta aplicació en C, s'ha fet servir l'aplicació Dev-C++, que inclou compilador de C. Per escriure el programa s'han consultat diferents fonts per saber com enviar dades pel port sèrie del PC. Cal dir que el port sèrie es tracta com un arxiu i la funció que s'utilitza per obrir i configurar-lo és:

```
CreateFile( pcCommPort,              //nom del port, per exemple "COM2"
GENERIC_READ | GENERIC_WRITE, // opció lectura/escriptura
0,                            // s'ha d'obrir amb accés exclusiu
NULL,                         // sense atributs de seguretat
OPEN_EXISTING,                // s'ha de fer servir OPEN_EXISTING
FILE_ATTRIBUTE_NORMAL,        // sense solapar I/O
NULL
);
```

Com es pot apreciar, és com un arxiu. Un cop s'ha obert amb èxit, resta configurar les opcions de comunicació. Seguint la configuració establerta per l'adaptador Bluetooth sèrie, la configuració serà la següent:

```
dcb.BaudRate = CBR_9600;     // ajustem el baud rate a 9600
dcb.ByteSize = 8;             // mida de les dades
dcb.Parity = NOPARITY;        // sense bit de paritat
dcb.StopBits = ONESTOPBIT;    // 1 bit de stop
```

Finalment, un cop obert el port i establerta aquesta configuració, podem escriure pel port sèrie amb la funció següent:

```
WriteFile(hCom, &buffer[i], 1, &written, NULL);
```

Aquesta funció escriu al "hCom", que és el fitxer creat anteriorment, el valor del caràcter del vector "buffer" que està en la posició i-èssima. Així, un per un s'envien al port sèrie tots els caràcters que formin les dades del *buffer*.

Fins aquí, s'ha parlat de com s'envien les dades pel port sèrie. A partir d'ara es tractarà el cos principal del programa, la funció *main*.

L'esquema del programa principal és aquest:

```
int main(int argc, char *argv[]) {
    numero = 0;
    index = 49;                          // índex al buffer
    vaciar_buffer();                     // buidem el buffer
    while (1) {                          // bucle infinit
        generar_num();                   // generem un número
        sleep(500);                      // esperem un cert temps
        obtener_cars();                  // el convertim a caràcters
        if (index <= 0) {                // si hem omplert el buffer
                  enviar();              // enviem les dades
                  vaciar_buffer();       // buidem el buffer
                  index = 49;            // reiniciem l'índex
                  sleep(5000);
        }
    }
}
```

Com es pot veure, el programa principal és senzill. El que fa és generar números, guardar-los a un *buffer* i quan aquest estigui ple, enviar-los pel port sèrie i després buidar-lo. El número generat és de tipus enter i s'ha de convertir a caràcters abans de ser enviat.

També és important esmentar que aquest programa escriu per pantalla les dades que envia, per després poder fer la comprovació de les dades que rep el dispositiu receptor.

3.3.3 PROGRAMA EN JAVA

S'han fet modificacions al programa en JAVA desenvolupat durant la primera fase del projecte per adaptar-lo a les necessitats d'aquesta segona fase. A la primera part, referent a la connexió Bluetooth, no hi ha canvis. Ara bé, es necessita conèixer la integritat de les dades que rep i per això, es va modificar el programa perquè mostrés per pantalla les dades que anava rebent.

L'aplicació en JAVA rep un vector de 50 caràcters. Entre aquests caràcters trobem números de 4 xifres separats per "," amb possibles zeros a l'esquerra. Així resulta més fàcil extreure els valors del vector que rep, ja que tenen una mida fixa. A més, per comprovar que tenim els números correctes calculem la mitjana aritmètica i es mostra per pantalla. Com es pot comprovar per part del programa en JAVA no hi ha grans canvis.

Com a visió general d'aquesta fase, al PC tenim el programa desenvolupat en C que envia les dades pel port sèrie i les mostra per pantalla. Està connectat al port sèrie del PC l'adaptador Bluetooth sèrie RS-232 que hem configurat prèviament i que enviarà les dades que rebi via Bluetooth al dispositiu que estigui connectat. Per últim, s'ha modificat l'aplicació desenvolupada durant la primera fase per poder estudiar i comprovar la integritat de les dades.

Finalment, hi ha èxit en la comunicació. Es connecta el telèfon mòbil a l'adaptador, llavors s'executa el programa escrit en C que genera i envia nombres aleatoris cada cert temps des del PC cap al mòbil, i aquest els rep i els tracta.

3.4 CONCLUSIÓ

Aquesta fase serveix per conèixer i configurar el dispositiu adaptador Bluetooth sèrie RS232 que connectarem al telèfon mòbil. A més ha estat útil per poder conèixer amb molta més precisió com funciona la comunicació via sèrie i quins han de ser els seus paràmetres de configuració. També, s'han desenvolupat funcions per a l'aplicació en JAVA que descodifiquen els valors que rep, que es faran servir més endavant a l'aplicació final.

La següent fase del projecte consistirà en substituir el PC i el seu programa en C per la placa de desenvolupament al simulador Proteus que emularà els sensors i enviarà les dades pel port sèrie.

CAPÍTOL 4

DESENVOLUPAMENT DEL FIRMWARE AL SIMULADOR PROTEUS

4.1 OBJECTIU

L'objectiu d'aquesta fase és desenvolupar mitjançant el simulador Proteus el codi que es farà servir a la placa de desenvolupament. Així, un cop desenvolupat aquest programa, només s'haurà de descarregar a la placa real i potser fer alguns canvis. Es preferible utilitzar software de simulació que no pas desenvolupar sobre la mateixa placa ja que dóna més facilitat per fer canvis, provar diferents configuracions, i sobretot per la possibilitat de comprovar el funcionament del programa mentre està en marxa. A més, com s'ha esmentat durant la introducció, evitem haver d'utilitzar la placa durant molts dies, ja que és propietat de la FIB.

4.2 ENTORN

Durant aquesta nova fase el hardware és igual i s'ha fet servir nou software. Per part d'aquest últim, s'ha instal·lat l'entorn MPLAB de Microchip v7.20 per escriure el codi i compilar-lo. Per això, s'ha instal·lat el compilador de C anomenat CCS amb el plugin[15] per què es pugui fer servir amb MPLAB. Finalment, s'ha instal·lat Proteus v7 Professional i s'ha fet servir l'esquema de la placa de desenvolupament.

4.3 DESCRIPCIÓ

Els canvis a nivell hardware que s'indiquen en aquesta fase fan referència a la placa de desenvolupament real, però es realitzen de manera molt semblant al model de la placa per a Proteus.

4.3.1 PROGRAMA DE PROVA

Com s'ha esmentat al principi, es tracta de desenvolupar amb el simulador proteus el programa que ha de dur la placa de desenvolupament, és a dir, la bicicleta. Primer de tot, i per comprovar la correcta configuració de l'entorn, s'ha utilitzat un programa senzill escrit en C que es compila i es carrega al simulador.

Com a programa d'exemple s'ha fet servir el següent:

```
#include "16F876_CCS.h"
extern void main( void)
{
PORTB = 0b00000000;
ADCON1 = 0b00000110;
TRISB = 0;
TRISA = 0b00111111; /* xxxx 0001 */
for(;;)PORTB=PORTA;
}
```

Al fitxer 16F876_CCS.h hi ha la declaració de les adreces de tots els registres del microcontrolador i dels bits que el conformen. Aquest senzill programa copia el valor del port A al port B. A efectes del funcionament a la placa, això vol dir que quan els interruptors s'obren els leds s'encenen, i quan es tanquen els leds s'apaguen. Els interruptors són el port A i els leds el port B. Es fan proves al simulador i el funcionament és correcte.

Un cop s'ha comprovat que es pot escriure, compilar i executar programes correctament a l'entorn, s'ha escrit el firmware que durà la placa de desenvolupament.

Aquest firmware serà com el programa de control de la bicicleta, que mesura les dades dels sensors i les envia al dispositiu mòbil. En aquest cas, es faran servir els diversos elements que conté la placa de desenvolupament per generar les dades dels sensors, el microcontrolador les captarà i les enviarà pel port sèrie on està connectat l'adaptador Bluetooth sèrie RS232.

Com a elements per generar les dades s'ha fet servir un generador de tensió variable en funció del recorregut del potenciòmetre, i un generador de polsos d'ona quadrada asimètrica, amb freqüència d'oscil·lació de 1 a 150 Hz. El primer es fa servir per emular el sensor que capta l'esforç que realitza l'usuari, mentre que el segon emula un cardiòmetre.

4.3.2 FUNCIÓ PRINCIPAL

Aquesta tercera fase s'explicarà a partir del bucle principal del programa:

```
extern void main()
{
  while (1) {                 //bucle infinit
            força();          //Adquirim l'esforç i l'enviem
            espera(2000);
            pols();           //Adquirim el pols i l'enviem
            espera(2000);
  }
}
```

Com es pot veure, es tracta d'un codi molt senzill i basat en el programa en C desenvolupat a la fase anterior. Cal dir que aquest és un esquema del que seria el programa real. Aquest esquema no mostra com el microcontrolador passa a l'estat "sleep[17]" a l'espera de que hi hagi dades. En un projecte com aquest el consum és un valor afegit i no resulta òptim un bucle infinit.

Mentre hi hagi aquest programa carregat a la memòria del microcontrolador sempre estarà realitzant aquestes operacions ja que s'emmarquen en un bucle infinit. El que fa aquest programa principal és obtenir els valors de d'esforç provinents del generador analògic de la placa de desenvolupament i quan estigui el *buffer* ple, enviar-los per la línia sèrie cap a l'adaptador Bluetooth. A continuació, esperarà durant un petit espai de temps, 2 segons, i realitzarà la mateixa operació però aquest cop amb el valor de les pulsacions.

El programa ha d'esperar durant un cert temps ja que no interessa saber l'esforç o les pulsacions de l'usuari en tot moment, amb conèixer el seu valor cada 5 segons és més que suficient. S'ha de tenir en compte que un exercici pot durar hores i es captarien moltes dades redundants.

Un cop vist el programa principal, a continuació es detalla cadascuna de les parts que el componen, com s'obtenen els valors i com s'envien a través del port sèrie RS232 de la placa de desenvolupament.

4.3.3 ALTRES FUNCIONS

La funció força() del cos del programa principal s'encarrega de captar i enviar les dades tal com s'ha explicat abans. El seu codi és el següent:

```
void força() {
 char enviat = 0;
 index = 49;
 buidar_buffer();
       while (!enviat) {
            adquirir_força();
            obtenir_cars();
            if (index <= 0) {
                  enviar();
                  buidar_buffer();
                  enviat = 1;
            }
       }
}
```

En aquest programa tant els valors de l'esforç com els valors de les pulsacions es guarden en el mateix *buffer*, per això al principi d'aquesta funció es buida. A continuació s'entra al bucle de la funció que, mentre no s'ha omplert el *buffer* amb les dades i s'ha enviat per la línia sèrie, no finalitza. A dins del bucle el funcionament és senzill. Amb la funció adquirir_força() obtenim un valor decimal de l'esforç que està fent l'usuari. Un cop es té aquest valor decimal, amb la funció obtenir_cars() es converteix a caràcters per ser enviat pel port sèrie i s'insereix al *buffer*. Si el *buffer* és ple (index = 0) s'envia pel port sèrie amb la funció enviar(). Aquesta funció és genèrica i s'utilitza per enviar els valors de l'esforç i de les pulsacions, però això s'explicarà més endavant. Seguint amb el transcurs d'aquesta funció, després d'enviar les dades, es buida el *buffer* i s'indica al bucle principal que s'ha enviat posant el valor de la variable enviat a 1. Finalitza el bucle i acaba la funció.

4.3.4 CONFIGURACIÓ DEL GENERADOR ANALÒGIC

Pel correcte funcionament de l'aplicació i d'aquesta funció en concret, cal establir diverses configuracions tant a nivell físic de la placa de desenvolupament com a nivell software del microcontrolador.

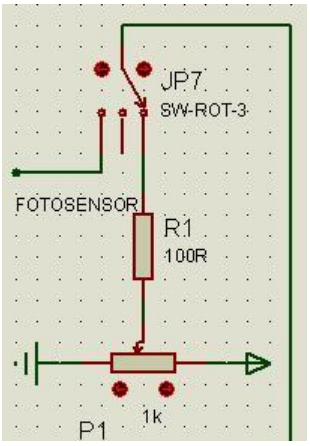

FIGURA 12: Esquema del generador de polsos

Per part de la configuració física del generador analògic, s'han d'establir alguns dels *jumpers* situats a la placa. Hi ha un *jumper* (el JP7, que es pot veure a la imatge) que separa els dos circuits que tenen generadors analògics: el fototransistor i el potenciòmetre. Hem d'ajustar el *jumper* per què el circuit seleccionat sigui el potenciòmetre, és a dir, el VA2.

A més, a la placa hem de fer que l'entrada RA0/AN0 del microcontrolador sigui el senyal provinent d'aquest circuit, així que també hem d'ajustar el *jumper* JP5 seleccionant AN0.

Per poder captar amb el microcontrolador aquesta tensió generada pel circuit de la placa, s'utilitza el convertidor analògic-a-digital que porta integrat el PIC16F876. Aquest convertidor genera un resultat digital de 10 dígits a partir d'un valor analògic a través d'aproximacions successives. Per aconseguir-ho, el mòdul convertidor pren com a tensió referència l'entrada VDD, Vss, RA2 o RA3, que són seleccionables per software.

Un cop s'ha configurat la placa de desenvolupament per què les senyals procedents dels perifèrics cap al microcontrolador siguin les correctes, cal configurar per software els registres del convertidor del microcontrolador. Aquesta configuració es duu a terme a la funció adquirir_força(), els registres involucrats en la conversió són els següents:

• A/D Registre de resultat alt (ADRESH)

• A/D Registre de resultat baix (ADRESL)

• A/D Registre de control 0 (ADCON0)

• A/D Registre de control 1 (ADCON1)

• PORTA Registre que determina la direcció de les dades del port A (TRISA)

El parell de registres ADRESH:ADRESL contenen els 10 bits resultat de la darrera conversió completa. Aquest registre, però, té d'amplada 16 bits (8 bits per cada part) que dóna la flexibilitat de justificar a dreta o esquerra els 10 bits de resultat, segons com es configuri al registre de control. Els bits extra per arribar a completar el parell són ajustats amb el valor '0'. Aquests dos registres no s'han de configurar ja que només s'han de llegir per obtenir el valor de la conversió.

El registre de control 0 configura diversos aspectes de la conversió que es veuen a continuació:

bit 0 ADON: bit A/D en marxa.

1 = Mòdul convertidor A/D està operatiu

0 = Mòdul convertidor A/D està apagat

bit 2 GO/DONE: estat de la conversió

1 = Conversió en progrés

0 = No està convertint

bit 5 - 3 CHS2:CSH0: bits de selecció de canal

000 = canal 0 (AN0/RA0)

bit 7 - 6 ADCS1:ADSC0: bits de selecció del rellotge de conversió

10 = 32Tosc 20 MHz

El bit 0 ens permet posar en marxa el convertidor mentre que el bit 2 ens informa de l'estat de la conversió. Aquest bit permet fer programes que funcionin mitjançant *main* en comptes de fer-ho per interrupció. Els bits del 3 al 5 seleccionen el canal d'entrada de la tensió a convertir. Amb el valor 000 s'habilita el canal d'entrada procedent de RA0/AN0, que ha sigut seleccionat com s'ha explicat abans amb el *jumper*[16] JP5, obtenint el senyal procedent del generador de tensió variable.

Finalment per part d'aquest registre, els bits 6 i 7 ajusten la freqüència d'oscil·lació del rellotge a 20 MHz amb el valor 10.

Així, ajustarem el registre de control 0 amb el següent valor:

```
ADCON0 = 0b.1000.0001;
```

Pel que fa al registre de control 1, també configura dues opcions del convertidor; els bits de control de configuració del port i la justificació del resultat al parell de registres que s'han vist anteriorment.

A continuació la descripció del registre bit a bit:

bit 0 – 3 PCFG0:PCFG3: configuració dels bits de control del port

1110 = Tensió de referència VDD i VSS i AN0 com a entrada analògica

bit 7 ADFM: bit de selecció de format del resultat

1 = Justificació a la dreta. Els 6 bits més significatius de ADRESH són llegits com zeros.

0 = Justificació a l'esquerra. Els bits menys significatius de ADRESL són llegits com zeros.

Els bits del 4 al 6 no estan implementats i es llegeixen com zeros.

Els bits de control del port seleccionen les tensions de referència. Es poden seleccionar pins del port A com a referència, però en aquest cas no ha de ser així, per això cal prendre com a tensions de referència les pròpies del microcontrolador VDD i VSS.

Per últim, justificarem el resultat de la conversió a l'esquerra. Així tindrem els bits de més pes del resultat a ADRESH i els dos últims bits, els de menys pes a ADRESL.

Llavors, el registre de control 1 queda configurat de la següent manera:

```
ADCON1 = 0b.0000.1110;
```

L'últim registre configurat per fer la conversió és el registre TRISA. Aquest registre determina la direcció de les dades dels bits del PORT A. En aquest cas s'especifica que el bit 0 del registre TRISA sigui d'entrada, ja que l'objectiu és que tingui la tensió que volem mesurar.

Per tots aquests aspectes, es configura el registre TRISA aplicant una màscara de la següent manera:

```
TRISA = TRISA || 0b.0000.0001;
```

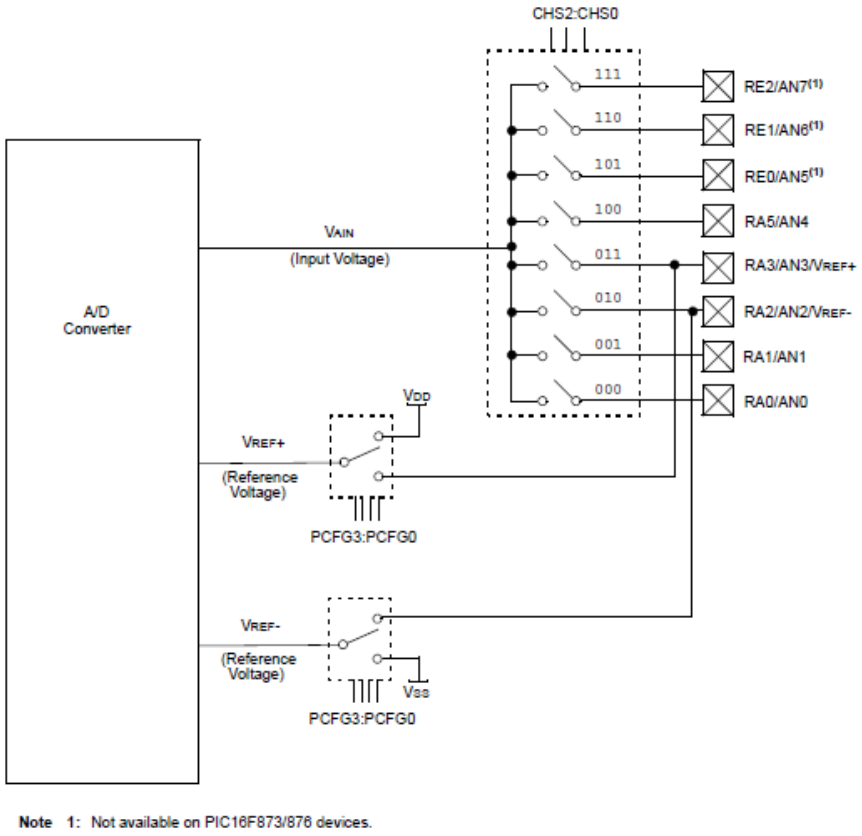

FIGURA 13: Esquema de les senyals de configuració del convertidor A/D

Fins aquest punt s'ha explicat com configurar els registres del convertidor analògic-a-digital. Ara cal fer una ullada al seu funcionament. S'ha escollit fer la conversió mitjançant el sistema d'interrupcions, tot i que es podria haver utilitzat el sistema *main* esmentat anteriorment. Els passos que s'han de seguir per dur a terme una conversió A/D per interrupcions són els següents:

1. Configuració dels registres de control del convertidor A/D, ADCON0, ADCON1, amb els valors explicats anteriorment.
2. Configuració de la interrupció:

   - Posar a 0 el bit ADIF

   - Posar a 1 els bits ADIE, PEIE, GIE

3. Esperem durant els temps d'adquisició del valor d'entrada.
4. Comencem la conversió posant a 1 el valor del bit GO/DONE, (GO = 1)
5. Esperem a que es completi conversió, esperant la interrupció del A/D.

6. Llegim els registres ADRESH:ADRESL que contenen el resultat tenint en compte la justificació.
7. Posem a 0 el bit ADIF i tornem a començar el procés.

En aquest procés de conversió entren en joc alguns bits dels quals no hem parlat i que tenen a veure amb la programació per interrupcions del microcontrolador.

Per configurar la interrupció:

- Registre PIR1 <6> 0 (ADIF)
- Registre PIE1 <6> 1 (ADIE)
- Registre INTCON <7.6> 11 (PEIE,GIE)

Fins ara, s'ha tractat el generador de tensió analògica que s'utilitza per emular l'esforç que realitza l'usuari. Girant el potenciòmetre de la placa de desenvolupament variarem la tensió de sortida del circuit i així anirem generant valors que simbolitzaran les mesures que prendria el sensor i que s'envien pel port sèrie cap a l'adaptador Bluetooth. Com s'ha explicat abans, totes les configuracions i processos per aconseguir captar la tensió es realitzen a la funció adquirir_força().

A continuació s'explica de forma semblant la configuració i la forma de captar els valors de l'element que s'utilitza per emular el cardiòmetre, que tal i com s'ha esmentat al començament d'aquesta fase, és el generador de polsos de la placa de desenvolupament.

4.3.5 CONFIGURACIÓ DEL GENERADOR DE POLSOS

El codi és simètric a la funció que capta, i envia l'esforç de l'usuari aplicant, evidentment, els canvis oportuns per adaptar-lo al generador lògic. No és necessari explicar la funció pols() ja que és igual, canviant a dins la funció adquirir_força() per adquirir_pols().

El generador lògic de la placa de desenvolupament es tracta d'un circuit construït en torn al Timer 555[18], que proporciona un senyal d'ona quadrada asimètrica amb una freqüència ajustable entre 1 i 170 Hz aproximadament. Mitjançant el recorregut d'aquest potenciòmetre es pot manipular l'amplada entre els polsos, i amb el firmware mesurar la seva amplada o el període. Així s'emulen els batecs del cor de l'usuari per captar-les amb el microcontrolador i més tard enviar-les via port sèrie RS232. Com es pot veure a la imatge, al costat del *timer* hi ha un díode led que mostra una idea visual de la freqüència de sortida del circuit. Igual que abans, per poder captar la senyal provinent del generador, hem d'aplicar canvis físics i lògics.

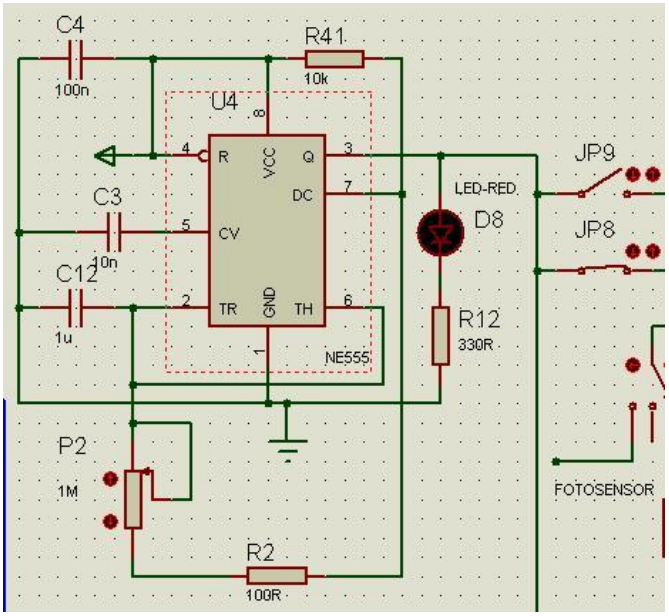

FIGURA 14: Esquema dels generadors de polsos

S'han d'utilitzar els *jumpers* de que disposa la placa de desenvolupament. Per mesurar l'amplada entre els polsos es fa servir el Timer 1. El Timer 1 pot funcionar com a comptador de polsos d'un rellotge extern, que en aquest cas serà el generador d'ona de la placa de desenvolupament. Per connectar la sortida del circuit generador lògic amb el Timer 1 del microcontrolador es tanca el *jumper* JP8 que es pot veure a la imatge anterior.

El mòdul del Timer 1 genera una interrupció quan el parell de registres TMR1H:TMR1L s'incrementa del valor hexadecimal FFFFh a 0000h. Com s'ha dit anteriorment, compta els polsos provinents del generador lògic, així que quan es genera la interrupció es comença a comptar l'amplada fins que arribi el següent pols. Així obtindrem el valor del temps que hi ha entre polsos i podrem conèixer la freqüència. Els registres relacionats amb el Timer 1 com a comptador de polsos externs:

• TMR1L: Registre amb els bits menys significants del *timer*.

• TMR1H: Registre amb els bits més significants del *Timer*.

• T1CON: Registre de control del Timer 1.

Igual que al convertidor analògic-a-digital, tenim un parell de registres que emmagatzemen el valor del Timer 1. Aquest registre és el parell format per TMR1H:TMR1L. El valor d'aquest registre s'incrementa segons els flancs del rellotge i també segons el valor del *preescaler*[19] . Per una altra banda, el registre T1CON configura diferents paràmetres del *Timer*. Es veu amb precisió a continuació:

bit 0 TMR1ON: bit de posada en marxa del *Timer*.

1 = Habilita el Timer 1

0 = Atura el Timer 1

bit 1 TMR1CS: bit de selecció de la font del senyal de rellotge

1 = Rellotge extern provinent de T1CKI

0 = Rellotge intern Fosc/4

bit 2 T1SYNC: bit de control de la sincronització amb el rellotge extern

1 = No es sincronitza amb l'entrada del rellotge extern

0 = Es sincronitza

bit 3 T1OSCEN: bit de control per habilitar l'oscil·lador del Timer 1

1 = L'oscil·lador està habilitat

0 = L'oscil·lador esta deshabilitat (s'apaga per evitar la pèrdua d'energia)

bit 4 – 5 T1CKPS0:T1CKPS1: bits de selecció del *preescaler*

00 = Valor 1:1 del *preescaler*

Els bits 6 i 7 no estan implementats i es llegeixen com un zero.

L'objectiu és comptar el temps que hi ha entre dos polsos. Ajustem al registre de control el bit 1 a 1 perquè el senyal de rellotge sigui extern, provinent del generador lògic de la placa de desenvolupament. Aquest senyal no el sincronitzarem amb les fases internes del Timer 1 ja que utilitzarem el Timer 0 per comptar el temps que passa entre polsos. Així el bit T1SYNC valdrà 0. Finalment deshabilitarem l'oscil·lador del Timer 1 i seleccionarem el *preescaler* 1:1, ja que l'amplada entre els polsos a 1 Hz és molt gran, i l'amplada entre els polsos a 170 Hz bastant petita. Si ampliéssim o reduíssim aquest temps podríem tenir problemes de sobreeiximent al Timer comptador, el Timer 0.

El registre de control queda amb aquest valor:

```
T1CON = 0b.0000.0111;
```

El funcionament per captar el temps entre dos polsos provinents del generador cap al Timer 1 és el següent:

1. Es configura el registre de control T1CON amb els valors esmentats anteriorment.
2. Configuració de la interrupció:
   - Posar a 0 el bit TMR1IF
   - Posar a 1 els bits TMR1IE, PEIE i GIE
3. S'ajusta el valor del parell TMR1H:TMR1L a FFFFh per provocar la interrupció al pròxim flanc de rellotge.
4. Esperem a l' interrupció provocada pel Timer 1.
5. Un cop executem la rutina de servei a la interrupció provocada pel Timer 1, sabem que ha arribat un pols des del rellotge extern. Llavors, hem de configurar un altre cop el Timer 1 per rebre el segon pols i comencem a comptar el temps.
6. Esperem un altre cop per a la segona interrupció que indicarà l'arribada del segon flanc de rellotge. S'executarà la rutina d'interrupció i llavors aturarem el Timer que estava comptant. Així, obtindrem el valor del temps que ha transcorregut entre flancs de rellotge extern.
7. Posem a 0 el bit ADIF i comencem de nou el procés si ho necessitem.

Amb aquest procés s'obté el valor que simbolitzarà les pulsacions per minut de l'usuari de la bicicleta. Evidentment, tractarem per software que el rang no sigui entre 1 i 170, ja que una persona mai arriba a 1 pulsació per minut. Igual que abans, girant el potenciòmetre de la placa de desenvolupament variarem la freqüència de sortida del circuit i podrem emular la variació en les pulsacions de l'usuari durant un exercici.

Un cop ja es disposa al *buffer* de totes les dades que ens proporcionen els diferents elements de la placa de desenvolupament, s'envien mitjançant la funció enviar() pel canal sèrie. Aquesta funció realitza la configuració dels registres relacionats amb l'enviament a través d'aquesta línia. No hi ha canvis a nivell físic de la placa de desenvolupament.

4.3.6 CONFIGURACIÓ DEL CANAL SÈRIE

A nivell hardware, la placa de desenvolupament disposa d'un circuit d'adaptació de nivells basat en el popular MAX232. A més d'un connector DB9 femella per poder connectar dispositius a la placa de desenvolupament com per exemple un PC, útil per descarregar el firmware al microcontrolador, o com en aquest projecte un adaptador Bluetooth – sèrie RS232. El funcionament resumit seria aquest:

Pel pin 3 del connector DB9 (TxD) es reben les dades que vénen del dispositiu extern, en aquest cas des de l'adaptador Bluetooth – canal sèrie. Per aquest projecte, però, no necessitarem comunicació en aquesta direcció, ja que només tindrem comunicació enviant les dades provinents dels generadors des del microcontrolador cap a l'adaptador. Per part del microcontrolador cap a l'adaptador, que és la part primordial d'aquest projecte, les dades es transmeten pel pin RC6/TX, que passen pel circuit MAX232 que les converteix a nivells TTL i arriben a l'adaptador Bluetooth pel pin 2 del connector DB9 (RxD).

Per poder transmetre dades amb el microcontrolador pel port sèrie de la placa de desenvolupament es fa servir el mòdul USART[20] del mateix. És un mòdul complex amb moltes opcions que es descriurà limitant l'abast a aquest projecte i les opcions escollides.

Aquest mòdul pot ser configurat com a *full duplex*[21] asíncron modificant el valor del registre de control i estat TXSTA. La comunicació *full duplex* implica la possibilitat de comunicar-se en les dues direccions simultàniament ja que el transmissor i el receptor són independents, encara que en aquest projecte l'objectiu és enviar només dades des del microcontrolador cap a l'adaptador.

El funcionament de la transmissió asíncrona es basa en el registre de desplaçament (TSR). Aquest registre obté les seves dades del *buffer* de transmissió, TXREG. El registre TXREG es carrega amb dades per software, és a dir, és el registre on carreguem els valors provinents dels generadors per què s'enviïn pel canal sèrie. El registre TSR no es carrega fins que l'STOP bit ha sigut transmès. Quan s'ha transmès aquest bit, el registre TSR es carrega amb noves dades provinents del *buffer*, del registre TXREG. Per saber quan s'ha transmès la dada i carregar-ne de noves cal fer *main* al bit TRMT, que indica si el TSR està buit o està encara ple.

Aquest mòdul del microcontrolador funciona a una velocitat determinada per a l'enviament de dades. La comunicació es realitza amb l'adaptador Bluetooth, que com s'ha explicat amb anterioritat, s'ha configurat per treballar a un *baud rate* igual a 9600. S'ha de fixar el mateix valor a tots dos extrems de la comunicació per què aquesta sigui correcta. Per fer això, el microcontrolador disposa de 2 registres per establir el seu *baud rate*: TXSTA i SPBRG.

Al registre TXSTA s'han d'informar els següents bits:

bit 2 BRGH: bit de selecció de *baud rate* d'alta velocitat.

1 = Alta velocitat

0 = Baixa velocitat

bit 4 SYNC: bit de selecció del mode de comunicació del mòdul USART

1 = Síncron

0 = Asíncron

Com s'ha dit abans, el mode de comunicació serà asíncron i es seleccionarà l'alta velocitat per reduir el percentatge d'error. L'error a baixa velocitat per 9600 es situa a 1,73% mentre que a alta velocitat és 0,16%. Així, amb aquesta elecció, l'equació que fixa el *baud rate* és la següent:

```
Baud Rate = Fosc/(16(X+1))          on X = valor del registre SPBRG
```

El segon registre de configuració és el SPBRG. Si es consulta el manual del microcontrolador es trobarà una taula que indica el *baud rate* generat amb els diferents valors de SPBRG i de Fosc determinats. En aquest cas la freqüència d'oscil·lació és de 20MHz i el valor indicat a la taula per al registre SPBRG per aconseguir un *baud rate* de 9600 és 129. El resultat de l'equació és 9615.

Per acabar amb la transmissió sèrie, es descriu el procés que segueix la funció enviar() del firmware desenvolupat:

1. Habilitar el port sèrie fixant el bit SPEN (serial port enable bit) a 1.
2. Inicialitzar el registre SPBRG a 129 per un *baud rate* de 9600.
3. Habilitar la comunicació asíncrona posant a 0 els bit SYNC i habilitar també l'alta velocitat amb el bit BRGH a 1.
4. Ajustar a 1 el bit TXEN per permetre la transmissió.
5. Comencem la transmissió carregant el caràcter al registre TXREG.
6. *Main* del bit TRMT que val 1 quan TSR esta buit. Llavors tornem a començar si queden dades per enviar.

Proteus disposa d'una eina molt útil anomenada Virtual Terminal Model. Aquesta eina s'ha de configurar per tenir els mateixos valors de comunicació que s'han anat veient al llarg de la memòria. S'ha de connectar aquest element a la línia de sortida del circuit del MAX232 tal com es veu en aquesta imatge:

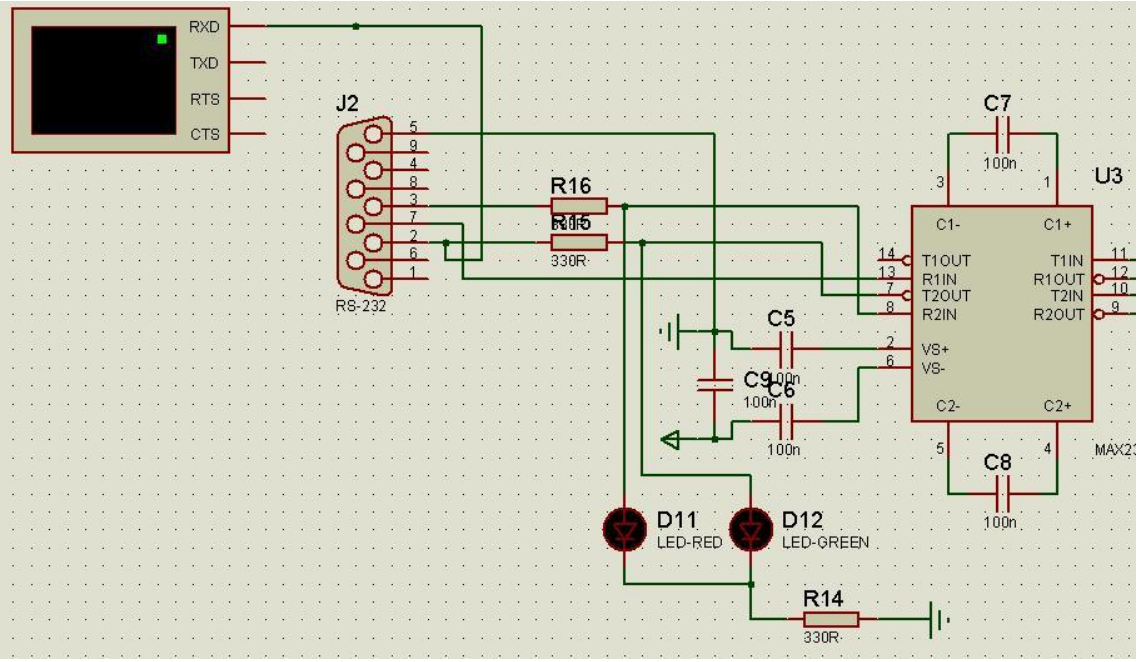

FIGURA 15: Esquema de les connexions entre el MAX-232, el connector DB9 i el terminal de sortida

Llavors, durant l'execució del programa es mostrarà una pantalla com la que es veu a continuació:

FIGURA 16: Imatge de la consola de sortida

En aquesta pantalla es mostren els caràcters que s'envien del microcontrolador pel canal sèrie. Així, podem examinar amb facilitat quines són les dades que es capten i s'envien i veure on pot haver un error.

4.4 CONCLUSIÓ

Finalment i com a conclusió d'aquesta tercera fase, cal remarcar que facilita l'adaptació a la placa de desenvolupament del firmware realitzat. Treballar amb el simulador Proteus ha sigut útil i ha permès fer moltes proves sobre el seu funcionament que hauria estat impossible fer-les a la placa directament. Ha sigut especialment útil l'últim element mostrat, ja que ha aportat molta llum al procés d'enviar les dades via sèrie.

Un cop feta la configuració de l'adaptador Bluetooth a sèrie RS232 i el firmware que durà la placa de desenvolupament, resta descarregar el programa a la placa i fer la connexió amb l'adaptador Bluetooth.

CAPÍTOL 5

ADAPTACIÓ DEL FIRMWARE I CONNEXIÓ ENTRE LA PLACA DE DESENVOLUPAMENT I L'ADAPTADOR BT

5.1 OBJECTIU

Durant la fase anterior ja s'ha desenvolupat el firmware amb el simulador Proteus que obtindrà els valors dels sensors i els enviarà pel canal sèrie de la placa. A l'altre extrem del canal sèrie trobem l'adaptador Bluetooth, que va ser configurat durant la segona fase, i enviarà aquestes dades cap al dispositiu mòbil de l'usuari.

Així, l'objectiu d'aquesta quarta fase és realitzar la connexió entre la placa de desenvolupament i l'adaptador Bluetooth i les primeres proves d'enviament de dades entre la placa i el terminal mòbil.

5.2 ENTORN

L'entorn d'aquesta quarta fase serà el laboratori. No és una fase per al desenvolupament sinó més aviat per a les proves. En aquestes s'utilitzarà el programari elaborat en altres fases, com l'aplicació per al dispositiu mòbil de la fase 1 i adaptada a la fase 2. Com s'explicava llavors, aquesta aplicació va ser desenvolupada principalment durant la fase 1, i per a la fase 2 es van fer modificacions per què mostrés per la pantalla del dispositiu els valors que rebia. Això ens ajudarà a examinar i estudiar els valors que rep de l'adaptador Bluetooth.

També es farà servir el programa "16F876A or PIC16F877A bootloader" per carregar el firmware al microcontrolador.

S'utilitzarà el hardware de les fases anteriors; un PC, la placa de desenvolupament, l'adaptador Bluetooth – sèrie RS-232 i un telèfon mòbil Sony-Ericsson K800i.

5.3 DESCRIPCIÓ

El primer pas de la fase és descarregar al microcontrolador el firmware desenvolupat. El microcontrolador de la placa ja té gravat un sistema operatiu anomenat PICMOS que ofereix la possibilitat de carregar programes del PC al microcontrolador a través del canal sèrie de comunicació. Això facilita desenvolupar aplicacions, fer proves i veure errors, per més tard canviar el codi i tornar a carregar l'aplicació. El software utilitzat per elaborar l'aplicació ja ens proporciona el fitxer amb el codi en hexadecimal, preparat per què el pugui executar el microcontrolador. Amb el programa *bootloader*[22] esmentat abans, es carrega el firmware sense problemes connectant entre el PC i la placa de desenvolupament el cable sèrie inclòs al paquet de la mateixa.

Automàticament podem comprovar que l'aplicació comença a funcionar a la placa de desenvolupament ja que hi ha un led de color verd que s'encén, i aquest led representa l'enviament de dades pel canal sèrie.

El següent pas és connectar la sortida del canal sèrie de la placa de desenvolupament amb l'adaptador Bluetooth. En aquest punt sorgeix un problema inesperat i és que els dos connectors DB9 de la placa i del dispositiu Bluetooth són femella i no es poden connectar directament amb el cable proporcionat esmentat abans.

La solució escollida va ser comprar un cable mascle – mascle als dos extrems. Un cop adquirit aquest cable, es van realitzar les primeres proves amb tot el sistema muntat, el firmware carregat, l'adaptador connectat al port sèrie de la placa de desenvolupament i el dispositiu mòbil connectat a l'adaptador.

El resultat va ser que el telèfon mòbil no rebia cap tipus de dades des de l'adaptador. Després d'estudiar el problema es va arribar a la conclusió de que la connexió no es realitzava correctament. Es va consultar documentació al respecte i el problema era degut a que els dos dispositius són del mateix tipus; DCE[23] . Per entendre el motiu s'ha d'aprofundir en la comunicació sèrie RS232. A continuació es veu el connector DB9 femella

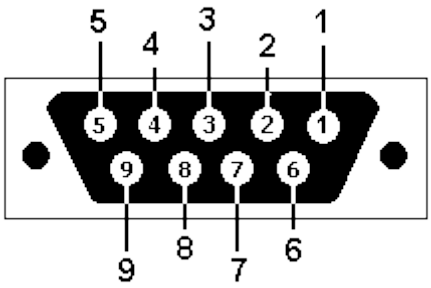

FIGURA 17: Connexions del connector DB9 Femella

Els pins elementals que es necessiten per aquest projecte són el 2, 3 i 5. El pin 2 és de sortida i és per on l'adaptador transmet les dades cap a l'altre dispositiu, el pin 3 és el pin de la recepció de dades i el 5 és el senyal de terra. Necessitem els pins de comunicació per transmetre dades i el pin número 5 per establir el mateix valor de referència de tensió per als dos dispositius.

El problema residia en què els dos dispositius transmetien i rebien les dades pels mateixos pins, i el cable sèrie RS232 mascle – mascle era paral·lel, és a dir, que enfrontava els pins de transmissió i de recepció.

Com a solució a aquest problema es va optar per fer el cable. Els pins 2 i 3 es creuen i ara la comunicació ja es pot establir com mostra aquest esquema:

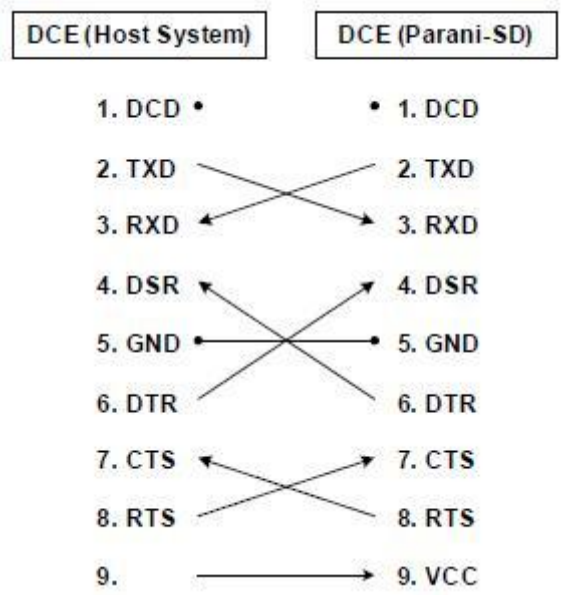

FIGURA 18: Esquema de connexió entre pins de dispositius DCE

Un cop carregat el firmware al microcontrolador, connectada la placa de desenvolupament a l'adaptador Bluetooth mitjançant el nou cable sèrie mascle – mascle, ja està llest el sistema per fer les primeres proves fent servir l'aplicació per al dispositiu mòbil desenvolupat en fases anteriors. L'aplicació del mòbil té la particularitat que mostra per pantalla totes les dades que rep. A continuació es descriuen els passos de la prova:

1. Carreguem, mitjançant el *bootloader*, el firmware de la fase anterior a la placa de desenvolupament.
2. Connectem fent servir el nou cable sèrie mascle - mascle l'adaptador Bluetooth amb la placa de desenvolupament.

3. Amb el dispositiu mòbil, executem l'aplicació elaborada durant les primeres fases del projecte. Ens connectem per Bluetooth al dispositiu trobat que es correspon amb l'antena que hem connectat i llavors es començaran a mostrar les dades que envia per pantalla .

Finalment, s'observa com el dispositiu mòbil mostra dades per la pantalla. Així que hi ha èxit en la comunicació.

5.4 CONCLUSIÓ

Aquesta fase ha sigut diferent a les anteriors ja que no ha sigut un treball de desenvolupament amb l'ordinador. No obstant, no ha estat exempta de problemes com s'ha pogut veure. Aquesta fase dóna cos al projecte; existeix comunicació entre les dues parts, la bicicleta, emulada per la placa de desenvolupament, i el dispositiu mòbil de l'usuari.

Finalment, i ja encarant la darrera part del projecte, resta desenvolupar una aplicació del dispositiu mòbil que obtingui les dades i les tracti.

CAPÍTOL 6

DESENVOLUPAMENT DE L'APLICACIÓ PEL DISPOSITIU MÒBIL

6.1 OBJECTIU

En aquesta cinquena i darrera fase, l'objectiu és crear el software del dispositiu mòbil que rebrà les dades i les tractarà. L'aplicació ha de cobrir els següents punts:

- Connectar-se a l'adaptador Bluetooth – sèrie RS-232.
- Mostrar per pantalla les dades que rep de l'adaptador, és a dir, l'esforç i les pulsacions i anar actualitzant-les en temps real.
- Desar en un log les dades que ha rebut.
- Mostrar per pantalla un resum i un gràfic orientatiu de l'exercici realitzat per l'usuari, on mostri les variacions a les pulsacions i l'esforç realitzat.

Així, l'objectiu és comunicar el dispositiu mòbil de l'usuari amb la bicicleta i tenir una eina que enregistri les dades i posteriorment ofereixi un tractament senzill de les mateixes.

6.2 ENTORN

L'entorn de desenvolupament utilitzat durant aquesta fase comparteix algunes aplicacions vistes anteriorment com l'emulador del WTK. En canvi, com l'aplicació creix considerablement de mida, s'ha decidit utilitzar un IDE[24] molt conegut de JAVA, Eclipse SDK 3.5.0. A més, s'ha instal·lat el plugin per a J2ME que incorpora accions per construir el projecte directament sense necessitat d'utilitzar la línia de comandes com a les fases prèvies.

Per fer les proves s'utilitzarà el mateix hardware de les fases prèvies.

6.3 DESCRIPCIÓ

Per poder cobrir les necessitats esmentades a l'apartat objectiu, s'ha dividit l'aplicació en diverses classes: Principal, Exercici, Conexio_BT, Arxiu, Grafic i Tupla. El codi d'aquestes classes es pot trobar al cd adjunt, no obstant, a continuació farem una descripció de cadascuna fent especial èmfasis en el seu paper a l'aplicació. Aquest és el diagrama de flux que segueix el programa:

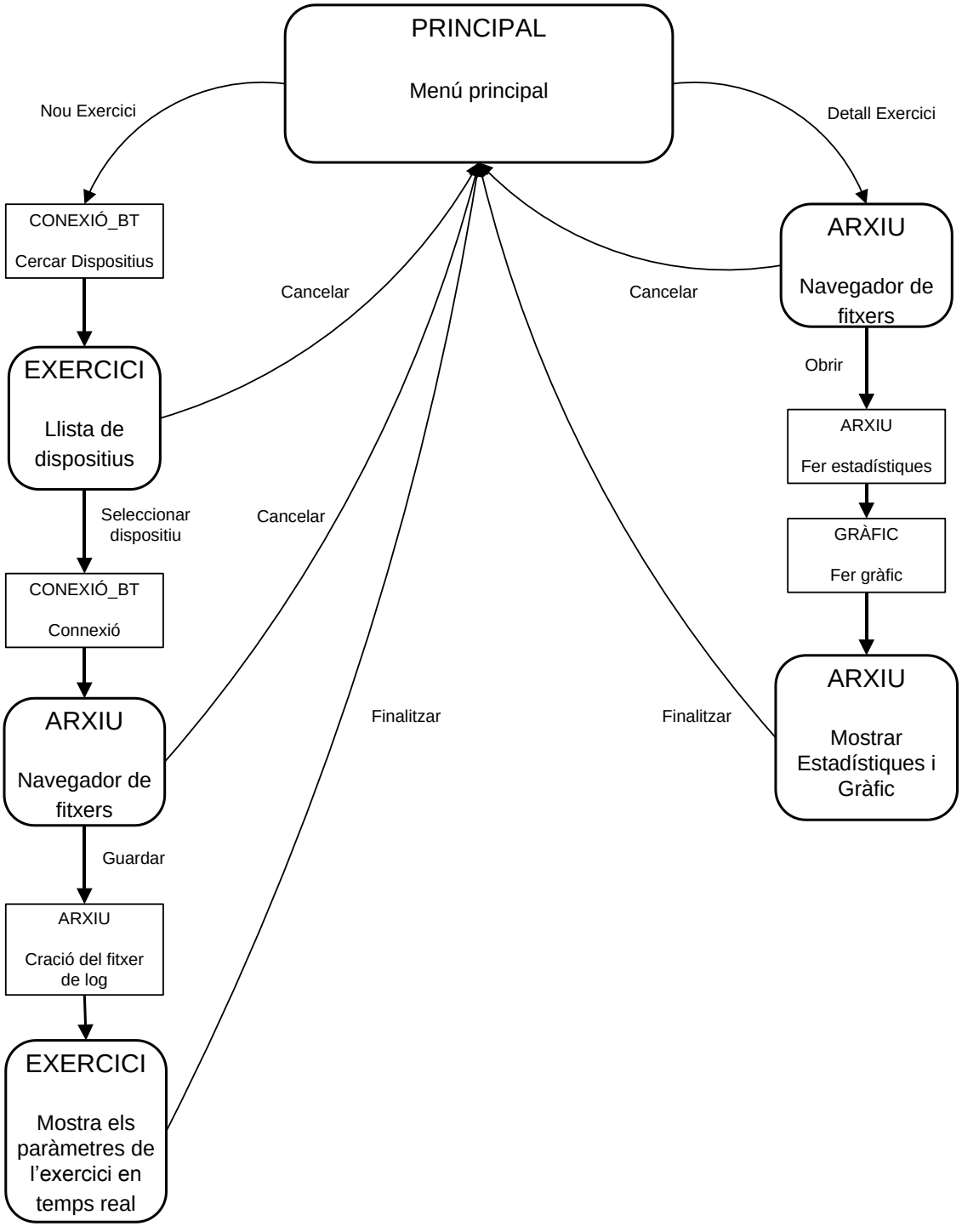

FIGURA 19: Diagrama de flux de l'aplicació del dispositiu mòbil

6.3.1 CLASSE PRINCIPAL

És la classe que estén la classe MIDlet, que són les aplicacions per a dispositius desenvolupades amb la plataforma J2ME. Representa el menú inicial i les accions principals. Aquest menú ofereix la possibilitat d'escollir entre l'opció de fer un exercici nou o examinar els resultats d'un exercici fet prèviament.

Si l'usuari escull l'opció de fer un exercici nou es crida a la funció següent:

```
public void exercici_nou() {
      if (conexio_establerta) {
            tancar_conexions();
      }
      exercici = new Exercici(this);
      if (!exercici.start()) {
            alerta("No s'han trobat dispositius");
            tornar_inici();
      }
}
```

Si s'ha establert alguna connexió prèviament, per exemple si es fan dos exercicis consecutius, es tanca la connexió i es crea una de nova. Llavors es crea un exercici nou cridant al constructor de la classe Exercici. El control del flux del programa passa a la classe Exercici.

Si en canvi s'escull l'opció d'examinar un exercici la funció que s'executa és aquesta:

```
public void examinar_arxiu() {
      arxiu = new Arxiu(this, null);
      alerta("Selecciona un arxiu per examinar");
      arxiu.start();
}
```

De manera semblant el control del programa passa a una altra classe, en aquest cas a la classe Arxiu.

Com s'ha vist, la classe Principal serveix de pont a les classes que representen cada acció que es pot donar al programa, fer un exercici nou o examinar un arxiu d'un exercici antic.

6.3.2 CLASSE EXERCICI

El programa es basa en exercicis i això és el que representa aquesta classe. Quan l'usuari escull l'opció de fer un nou exercici del menú principal, es crida al constructor de l'objecte d'aquesta classe. A partir del moment en què està creat, treballem amb aquest objecte per cridar a les funcions que afectaran a l'exercici.

El primer pas és establir la comunicació amb el dispositiu Bluetooth que transmetrà les dades. Aquesta part del codi és la que s'hereta de la primera fase del projecte. Per establir la connexió es crearà un nou objecte de la classe Conexio_BT. Aquesta classe es descriurà a continuació, però breument es pot dir que és la responsable d'establir la connexió amb l'adaptador Bluetooth.

Un cop establerta la connexió, es crea un arxiu de log on es guardaran totes les dades rebudes. Una altra classe anomenada Arxiu és la responsable de mostrar el navegador del sistema de fitxers, crear el fitxer i treballar amb ell.

A continuació es desarà l'arxiu i començarà la recepció de les dades. Si no es pot establir la connexió es mostra una alerta i es torna al menú inicial.

Com s'ha pogut comprovar, la classe Exercici utilitza la classe Conexio_BT i Arxiu per connectar-se amb l'adaptador Bluetooth i crear l'Arxiu de log que desarà les dades al dispositiu mòbil de l'usuari.

6.3.3 CLASSE CONEXIO_BT

S'utilitza, tal com s'ha esmentat, des de la classe Exercici. La classe Conexio_BT mitjançant threads[25] s'encarrega de buscar dispositius i els seus serveis. Aquesta llista de dispositius es retorna a la classe Exercici, que com s'ha vist abans, la mostra per pantalla i és l'usuari qui escull el dispositiu al qual es vol connectar. Quan l'usuari vol connectar-.se a l'adaptador, aquesta classe és l'encarregada d'establir la connexió i obtenir el canal d'entrada per on arribaran les dades. Aquest canal d'entrada serà el que utilitzarà la classe Arxiu per llegir les dades que arriben i desar-les a l'arxiu de log.

Per descobrir dispositius Bluetooth, la classe Conexio_BT ha d'implementar la classe DiscoveryListener. Per això s'han d'implementar les funcions:

- `public void servicesDiscovered(int transID, ServiceRecord[] records)`

S'inicia quan durant la cerca es descobreix un servei d'un dispositiu. Aquesta funció l'afegirà a la llista de serveis trobats si comença per la cadena de caràcters btspp .

- `public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)`

Comença quan es descobreix un dispositiu durant la cerca. S'afegeix el seu nom a la llista de dispositius trobats que més tard s'enviarà a la classe Exercici per que la mostri per pantalla.

- `public void inquiryCompleted(int discType)`

Aquesta funció s'executa quan s'ha finalitzat la cerca de dispositius. Marca la cerca com completada i segueix el transcurs de l'aplicació.

- `public void serviceSearchCompleted(int transID, int respCode)`

Es crida quan s'ha finalitzat la cerca de serveis d'un dispositiu.

6.3.4 CLASSE ARXIU

La classe Arxiu es crida des de la classe Exercici i des de la classe Principal segons l'elecció de l'usuari al menú inicial.

Quan l'usuari escull l'opció de fer un nou exercici, la classe Exercici inicia la classe Arxiu quan s'ha establert la connexió amb l'altre dispositiu. Arxiu mostrarà el sistema de fitxers perquè l'usuari pugui navegar i desar el fitxer en la localització que vulgui. Un cop escollida la ruta de destí, es crea l'arxiu i es comença la recepció de dades de l'adaptador Bluetooth. Per a la recepció de les dades es fa servir el canal de comunicació obert durant la connexió per la classe Conexio_BT. Com a nom d'arxiu s'estableix la data del dia d'execució. Si s'executa més d'un cop en el mateix dia s'afegeix un número indicatiu de l'orde.

Quan la classe Arxiu és cridada a partir de la classe Principal, és perquè l'usuari ha escollit l'opció d'examinar un fitxer d'un exercici anterior. En aquest cas també es mostra un navegador del sistema de fitxers del dispositiu mòbil on l'usuari pot escollir l'arxiu que vol examinar. Un cop seleccionat, es llegeix l'arxiu i es creen les estadístiques d'aquest exercici. Per a cada paràmetre estudiat es calculen tres dades estadístiques: la mitjana aritmètica, el valor màxim i el valor mínim. A continuació es mostraran aquests valors i finalment una gràfica amb l'evolució dels valors durant l'exercici. Per crear aquesta gràfica s'utilitza la classe Gràfic, que s'explica a continuació.

6.3.5 CLASSE GRÀFIC

A la classe Gràfic hi ha les eines per dibuixar el gràfic a partir de les dades guardades als fitxers. Aquesta classe estén la classe Canvas de JAVA feta per dibuixar bàsicament primitives gràfiques. Com aquest projecte només pretén implementar una eina senzilla per mostrar un possible tractament de les dades, amb aquesta classe hi ha suficient, ja que permet dibuixar els eixos de coordenades i les línies de les gràfiques en diferents colors. Es podria dir que és una forma de dibuixar gràfics a baix nivell.

Les gràfiques es fan independentment de la resolució de la pantalla. És important independitzar la gràfica de la mida de la pantalla ja que els dispositius mòbils són molt variats i pot comportar molts problemes de visualització. Això s'ha aconseguit utilitzant les funcions:

```
ample = getWidth();
alt = getHeight();
```

Aquestes funcions donen el valor de les mides de la pantalla de cada dispositiu. A més, a l'hora de dibuixar el gràfic s'ha fet en base a aquests valors. El bucle de la funció per dibuixar la gràfica és el següent:

```
for (i = 0; i < v.length; i++) {
      increment = (float) v[i] * stepAlt;
      y2 = (int) a - (int)increment - 1 ;
      x2 = (int) (x + stepAmple);
      g.drawLine(x, y, x2, y2);
      x = x2;
      y = y2;
}
```

Prèviament a aquesta funció, es defineixen les variables `stepAlt` i `stepAmple`. `StepAlt` i `stepAmple` es calculen tenint en compte la mida dels eixos i la quantitat de valors, en el cas de l'eix X, i en la mida dels valors per a l'eix Y. Aquestes variables representen la distància entre dos punts del mateix eix. És a dir, el punt Y1 està a una distància stepAlt del punt Y2. Simètricament per als valors de l'eix X.

La rutina `dibuixarGrafica` rep un gràfic i un vector on hi ha els valors a representar. Com es dibuixen els eixos de coordenades, s'ha de començar a dibuixar els valors a partir de X = 11 i Y = alt – 10, on alt representa el valor de la mida de l'eix Y del dispositiu. A

continuació s'executa un bucle que posiciona cada valor en un punt començant per l'origen i avançant segons les variables `stepAlt` i `stepAmple`.

6.3.6 CLASSE TUPLA

Finalment la classe Tupla representa una estructura per contenir els valors resum del fitxer que llegim. Conté els valors de la mitja, el màxim i el mínim.

6.4 CONCLUSIONS

En el desenvolupament s'han trobat dificultats sobretot en la programació de *threads*. S'ha hagut de programar així per evitar els deadlocks[26], per exemple al cercar dispositius o serveis, o al crear un arxiu i escriure-hi. També s'han trobat dificultats al programar en JAVA. Feia bastant de temps que no programava en aquest llenguatge, però el període d'adaptació ha sigut ràpid. Tampoc havia programat mai en J2ME, una versió reduïda del JAVA jdk. No obstant, s'ha aprofundit bastant en l'ús del J2ME; com tracta connexions, *streams*, arxius, pantalles i gràfics entre d'altres ha resultat difícil.

Tot i així els resultats han estat satisfactoris i s'han cobert les necessitats que es tenien per objectius. S'ha desenvolupat una eina senzilla de cara a l'usuari que rep les dades i les tracta donant una informació que pot resultar-li molt útil.

CAPÍTOL 7

PLANIFICACIÓ I ANÀLISI ECONÒMICA DEL PROJECTE

7.1 PLANIFICACIÓ DEL PROJECTE

La planificació que es va fer inicialment del projecte no s'ha pogut complir amb total precisió degut a diversos motius. En un primer moment no es va valorar prou bé la dificultat d'algunes parts i va fer que es retardés la seva consecució. Especialment la darrera fase ha sigut problemàtica en aquest sentit, ja que s'han necessitat més dies dels previstos per realitzar l'aplicació. També s'ha trigat més temps de l'espera't en la realització de la memòria, on les vacances de Nadal han suposat una breu interrupció en la seva redacció.

La planificació final ha estat la següent:

Nombre de tarea	Duración	Comienzo	Fin
Delimitació dels objectius del projecte	5 días	09 mar	15 mar
FASE 1: Creació d'un client Bluetooth per al mòbil en JAVA	**20 días**	**16 mar**	**12 abr**
Cerca d'informació sobre comunicacions Bluetooth	2 días	16 mar	17 mar
Cerca d'informació sobre el desenvolupament d'aplicacions per dispositius mòbils	4 días	18 mar	23 mar
Preparació de l'entorn de desenvolupament d'aplicacions per a mòbils	2 días	24 mar	25 mar
Disseny de la aplicació Beta client i servidor Bluetooth	12 días	26 mar	12 abr
FASE 2: Comunicació entre el mòbil i el PC mitjançant l'adaptador BT	**20 días**	**15 abr**	**12 may**
Estudi i configuració de l'adaptador Bluetooth Parani SD200	3 días	15 abr	17 abr
Cerca d'informació sobre les transmissions a través del port serie del PC	17 días	16 abr	08 may
Disseny de l'aplicació pel PC per enviar dades al mòbil utilitzant l'adaptador Bluetooth	8 días	29 abr	08 may
Correcció de problemes de l'aplicació per al PC desenvolupada	3 días	06 may	10 may
Proves per establir la comunicació entre el PC i el mòbil	2 días	11 may	12 may
FASE 3: Desenvolupament del firmware al simulador PROTEUS	**50 días**	**13 may**	**21 jul**
Preparació de l'entorn al PC per desenvolupar el firmware de la Placa	10 días	13 may	26 may
Cerca d'informació i estudi de la Placa	27 días	28 may	05 jul
Disseny del firmware de la Placa	6 días	06 jul	13 jul
Desenvolupament del firmware de la placa de SDMI	6 días	14 jul	21 jul
FASE 4: Adaptació del firmware i connexió entre la placa i l'adaptador BT	**7 días**	**22 jul**	**30 jul**
Instal·lació del software Bootloader	2 días	22 jul	23 jul
Descàrrega del codi al microcontrolador	2 días	23 jul	24 jul
Proves de comunicació entre el dispositiu mòbil i la placa de desenvolupament	4 días	27 jul	30 jul
FASE 5: Desenvolupament de l'aplicació pel dispositiu mòbil	**42 días**	**03 ago**	**29 sep**
Cerca d'informació sobre el desenvolupament d'aplicacions per a mòbils amb J2ME	5 días	03 ago	07 ago
Disseny de l'aplicació per al mòbil	5 días	10 ago	14 ago
Implementació de l'aplicació per al mòbil amb totes les funcionalitats	32 días	17 ago	29 sep
Elaboració de la memòria	15 días	23 dic	12 ene

FIGURA 20: Taula de tasques del projecte

El diagrama de Gantt de les tasques anteriors és el següent:

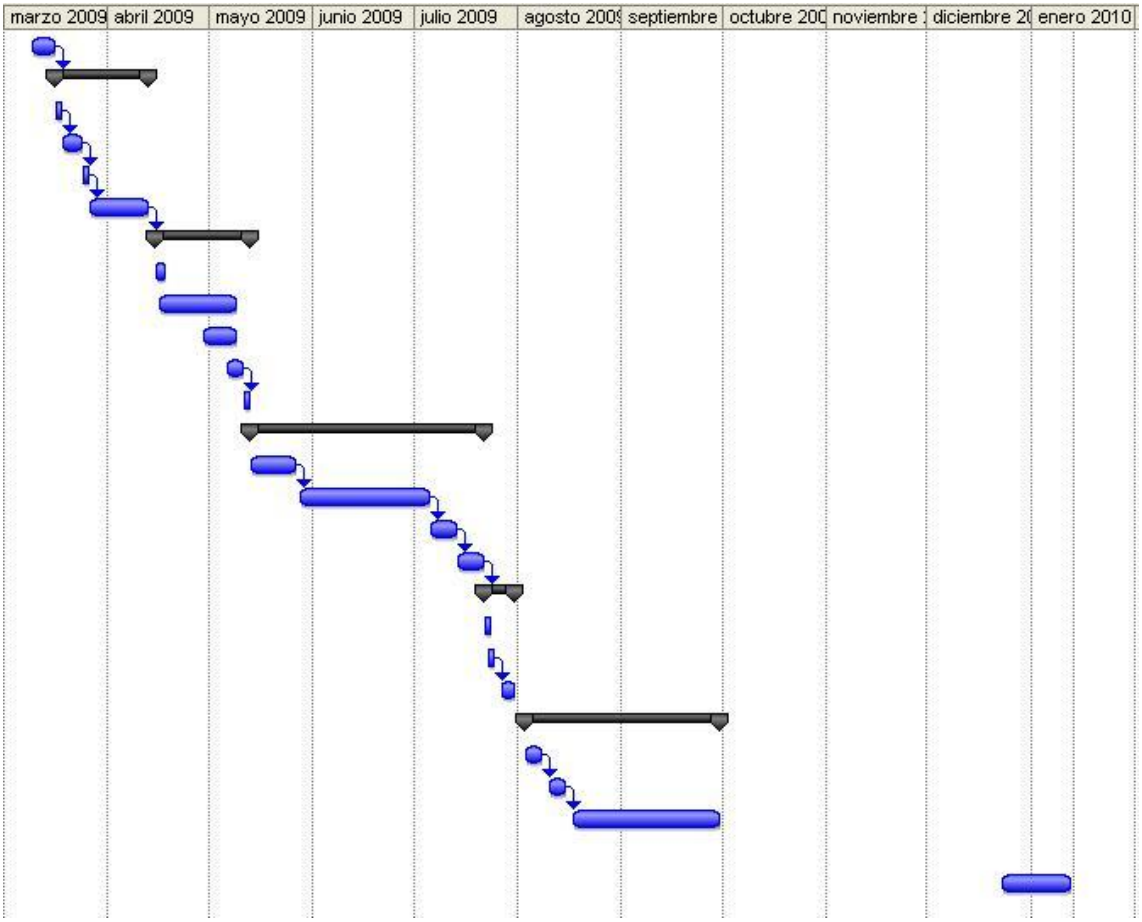

FIGURA 21: Dagrama de Gantt del projecte

Es pot apreciar que el projecte va començar a mitjans de Març de l'any 2009 i ha finalitzat a mitjans de Gener amb la realització de la memòria. Segons la planificació del projecte la darrera fase acabava just al començament de Setembre i durant els mesos lectius del quadrimestre s'aniria desenvolupant la memòria. Al final, durant aquests mesos no s'ha pogut treballar en el projecte i l'elaboració de la memòria s'ha realitzat entre finals de Desembre i principis de Gener.

7.2 ANÀLISI ECONÒMICA

Llistat del programari utilitzat:

Sun JAVA jdk 1.6	Gratuït
Sun JAVA Wireless Toolkit 2.5.2 for CLDC	Gratuït

Dev-C/C++	Lliure, llicència GPL
ParaniWIN	inclòs amb l'adaptador
Proteus 7 Professional	549.90 €
Microchip MPLab IDE V7.20	Gratuït
Eclipse IDE + plugin J2ME	Lliure, llicència EPL

Llistat dels equips:

Adaptador Bluetooth – sèrie RS-232 Parani-SD 200					68	€
Placa de desenvolupament PIC Laboratory MSE					86	€

Desenvolupament del projecte:

Anàlisis i disseny	110 hores	x	40 €/h	=	4400	€
Tècnic	300 hores	x	25 €/h	=	7500	€

COST TOTAL = 12.603,90 €

CAPÍTOL 8

CONCLUSIONS

Assoliment dels objectius

Finalitzat ja el projecte i revisant els objectius marcats al seu inici, es pot afirmar que s'han complert tots els objectius.

Gràcies a una planificació i divisió estructurada en fases s'han anat assolint els diferents objectius a mesura que avançava el projecte. Els dos elements finals del projecte que són el firmware i l'aplicació per al telèfon mòbil responen a les necessitats marcades a l'inici del projecte. S'han realitzat proves amb èxit d'exercicis emulats fent servir la placa de desenvolupament i variant amb els potenciòmetres les dades que generaven els sensors emulats. Aquestes dades eren captades pel microcontrolador i enviades a través del canal sèrie cap a l'adaptador. De l'adaptador eren transmeses cap al dispositiu mòbil on quedaven registrades en un arxiu de log al propi dispositiu. A continuació es pot examinar l'arxiu i l'aplicació mostra les estadístiques de l'exercici. Finalment, i tal i com es demanava a l'inici del projecte es mostra una gràfica amb l'evolució de les dades durant l'exercici.

Degut a les diferents tecnologies amb les que s'ha treballat i als problemes trobats s'ha hagut d'ampliar el temps previst per a la consecució del projecte. Imprevistos com l'enviament a través de l'adaptador Bluetooth – RS232, com el cable DB9 RS-232 mascle – mascle invertit o la complexitat de l'aplicació per al dispositiu mòbil han sigut en gran part responsables.

Aportació tecnològica

En aquest projecte intervenen moltes tecnologies diferents. Partint de codi a molt baix nivell, pràcticament utilitzant sentències assemblador, fins a codi d'alt nivell com és J2ME. S'ha programat en diferents llenguatges com són C i JAVA. A més s'ha treballat el hardware de la placa de desenvolupament, del microcontrolador i de l'adaptador Bluetooth – sèrie RS232.

També cal esmentar que s'ha treballat amb software molt variat com per exemple emuladors, simuladors, IDE, l'Hyper Terminal i línia de comandes. Tot i així no sempre ha estat

un treball de desenvolupament d'aplicacions i també s'ha fet treball més orientat al hardware durant la segona i quarta fase.

Durant cada fase s'ha treballat en entorns diferents i nou programari, per això l'aprenentatge ha sigut continu i en molts camps. S'ha de valorar que aquest sol projecte integra diversos camps de la informàtica com és la programació de microcontroladors, les comunicacions entre dispositius o la programació orientada a objectes en alt nivell.

Possibles ampliacions

Aquest és un projecte que pot seguir creixent en moltes direccions. És un bon punt de partida per assolir fites més grans, ja que estableix les bases per a la comunicació i el tractament de les dades.

Es podrien afegir nous sensors que mesuressin altres paràmetres de l'activitat. Es podrien afegir també càlculs per arribar a conèixer dades com el consum de calories per part de l'usuari durant l'exercici.

Una possible millora que ja s'ha esmentat l'inici d'aquesta memòria és deixar d'emular amb la placa de desenvolupament els sensors, i substituir-los pels sensors reals. Aquesta millora representarà molts reptes nous, tot i així, la part de comunicació està tota implementada en aquest projecte i pot ser de gran ajuda.

Una altra possible millora pot ser ampliar el tractament de les dades per part del dispositiu mòbil. Igual que en la millora proposada anteriorment, en aquest projecte s'estableixen certes bases que poden ser de gran utilitat, però la realització en sí d'aquesta millora pot ser tan complexa com es vulgui. Per exemple, es podria implementar un sistema d'accés a l'aplicació mitjançant usuaris, que permetria compartir la bicicleta sense conflictes entre les dades dels usuaris dins un mateix nucli familiar o dins d'un centre d'alt rendiment. També per part de l'aplicació del terminal, es podria millorar el gràfic i canviar-lo per un gràfic més estètic en 3D, fent servir classes més complexes dins de JAVA que no pas Canvas, que és a molt baix nivell.

Com s'ha esmentat durant la descripció de la fase tres, el bucle principal del firmware és ineficient en termes d'energia. Mentre no hi hagi dades disponibles del sensor s'ha de provocar un estat "sleep" per evitar el consum d'energia per part del microcontrolador. És important ja que la bicicleta fa servir bateries, i s'ha d'optimitzar l'ús de l'energia per prolongar la seva durada.

Com es pot apreciar, són múltiples les vies per a l'expansió i aprofundiment a partir d'aquest projecte final de carrera.

Valoració personal

Personalment he trobat molt atractiu aquest projecte des de l'inici. Sempre m'ha cridat l'atenció el hardware i en especial les assignatures del departament d'ESAII, i ha sigut una bona experiència tornar a programar la placa de desenvolupament que ja coneixia des de l'assignatura d'SDMI. A més, tot i que es pugui pensar que és un projecte eminentment dedicat al hardware i la seva configuració o creació del hardware, darrera hi ha moltes hores de programació en llenguatge d'alt nivell. L'aplicació del telèfon mòbil està formada per codi de més de 1500 línies. A més, feia bastant de temps que no programava i mai ho havia fet en J2ME, el que m'ha servit per conèixer una altra aplicació dins de tot el paradigma JAVA.

Així una de les claus del projecte ha estat la diversitat de programari, d'entorns i de tipus de treball que m'han impedit caure en la relaxació i m'han fet aprendre molt. He hagut de consultar moltes i diverses fonts d'informació de temes que mai havia tractat com els gràfics en JAVA, connexions Bluetooth o la comunicació via RS232.

Per finalitzar les conclusions, cal dir que estic molt satisfet de l'estructuració del projecte dissenyada en gran part amb el meu tutor del projecte, en Joan Climent. Ha estat molt important aquesta divisió per fases que ha servit per veure com mica en mica avança el projecte i no caure en el desànim a la espera de resultats que mai arriben.

També estic especialment satisfet d'haver aconseguit tots els resultats que al principi del projecte es veien molt llunyans i difícilment assolibles.

WEBGRAFIA

[1] Informació sobre la placa de desenvolupament *http://www-pagines.fib.upc.es/~sdmi/*

[2] Introducció a la comunicació Bluetooth *http://www.quatech.com/support/comm-over-bluetooth.php*

[3] Informació sobre l'estàndard Bluetooth *http://www.cs.utk.edu/~dasgupta/bluetooth/*

[4] Exemple de comunicació Bluetooth entre dispositius *http://www.codeguru.com/java/article.php/c13147__2/*

[5] Exemple d'aplicació per connexions Bluetooth *http://www.jsr82.com/jsr-82-sample-spp-server-and-client/*

[6] Informació sobre RS-232 *http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm*

[7] Cablejat RS-232 *http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html*

[8] Comunicació sèrie *http://www.taltech.com/TALtech_web/resources/intro-sc.html*

[9] APIs i documentació de J2ME *http://java.sun.com/javame/reference/index.jsp*

[10] Guia de creació d'aplicacions J2ME *http://today.java.net/pub/a/today/2005/02/09/j2me1.html?page=1*

[11] Guia de J2ME *http://leo.ugr.es/J2ME/MIDP/index2.htm*

[12] Curs de programació de dispositius mòbils *http://www.mailxmail.com/curso-programacion-juegos-moviles-j2me*

[13] Guia de gràfics en J2ME *http://leo.ugr.es/J2ME/MIDP/graficos.htm*

[14] Tractament d'arxius amb J2ME *http://developers.sun.com/mobility/apis/articles/fileconnection/*

[15] Exemple sobre llegir arxius en J2ME *http://roseindia.net/j2me/read-file.shtml*

[16] Navegador d'arxius J2ME *https://www6.software.ibm.com/developerworks/education/wi-navigate/section4.html*

[17] Exemples d'aplicacions en J2ME *http://www.java2s.com/Code/Java/J2ME/CatalogJ2ME.htm*

[18] Més exemples d'aplicacions J2ME *http://www.java-tips.org/java-me-tips/midp/*

[19] Wikipedia *http://en.wikipedia.org/wiki/Main_Page*

ANNEX A

MANUAL D'USUARI

A.1 REQUISITS MÍNIMS

El dispositiu mòbil ha de disposar d'intèrpret de JAVA i de Bluetooth v1.0.

A.2 INSTAL·LACIÓ DEL SOFTWARE AL DISPOSITIU MÒBIL

Si examinem el cd inclòs, hi ha dos fitxers executables. El primer fitxer és un fitxer .jar i l'altre és un .jad. Aquests arxius són els que s'han de transmetre al dispositiu mòbil. Depenent del dispositiu mòbil només farà falta el fitxer .jar o potser tots dos.

Un cop tinguem localitzats aquests fitxers al nostre PC o al CD-ROM, s'han d'enviar al telèfon mòbil. Hi ha vàries opcions:

1. A través de Bluetooth.
   Primer s'ha d'activar el Bluetooth de l'ordinador i del telèfon mòbil. Un cop activats, s'envien els fitxers esmentats anteriorment al dispositiu mòbil. Un cop finalitzi la transferència, depenent del tipus de telèfon mòbil, ens donarà l'opció automàticament d'instal·lar el software. Si és possible, és l'opció recomanada ja que l'instal·larà automàticament. Si per una altra banda no s'ofereix l'opció de la instal·lació automàtica, l'usuari haurà de buscar el fitxer executable al sistema de fitxers del dispositiu. Un cop el trobi, el telèfon oferirà l'opció d'instal·lar el software.

2. A través del cable de dades.
   Actualment, és comú que al paquet del telèfon mòbil s'inclogui el cable de dades. Aquest cable es connecta al USB del PC i permet transferir dades. Llavors, l'usuari ha de connectar el cable al dispositiu mòbil i al seu PC, navegar entre les carpetes del seu telèfon i desar els fitxers esmentats abans. S'ha de finalitzar la connexió de forma segura per no danyar les dades. Un cop transferits els arxius s'han d'instal·lar manualment, ja que el telèfon no oferirà l'opció de fer-ho de manera automàtica, al contrari que amb la transmissió Bluetooth. S'ha de buscar el fitxer executable dins del directori on s'ha guardat i executar l'opció instal·lar.

Un cop instal·lat, l'usuari ha d'anar al menú d'aplicacions del seu telèfon mòbil i ja el podrà executar.

A.3 FUNCIONAMENT DE L'APLICACIÓ

A.3.1 MENÚ INICIAL

En el menú inicial s'ofereixen dues opcions a l'usuari.

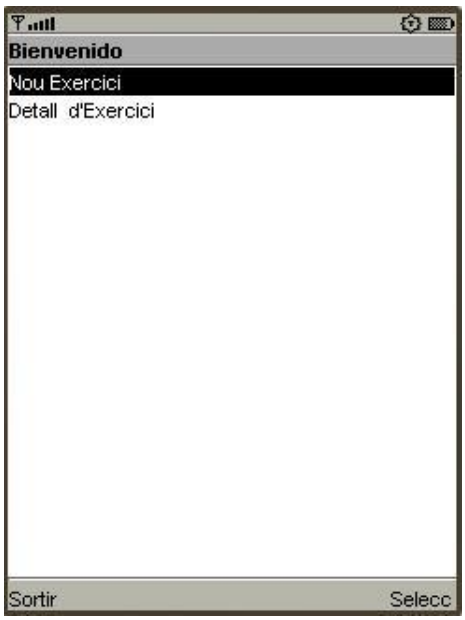

FIGURA 22: Menú principal

A.3.2 NOU EXERCICI

Nou exercici dóna l'opció a l'usuari de començar un nou exercici que registrarà les seves pulsacions i l'esforç que realitza. Aquest fitxer pot ser descarregat directament a un PC a través de Bluetooth o del cable de dades que faciliti el fabricant. Per saber com transferir arxius del telèfon al PC recomanem al lector la lectura del manual d'usuari del telèfon mòbil.

El primer pas és connectar-se via Bluetooth a la bicicleta. Per connectar-se, al triar l'opció de Nou exercici, l'aplicació cercarà dispositius propers i mostrarà una llista.

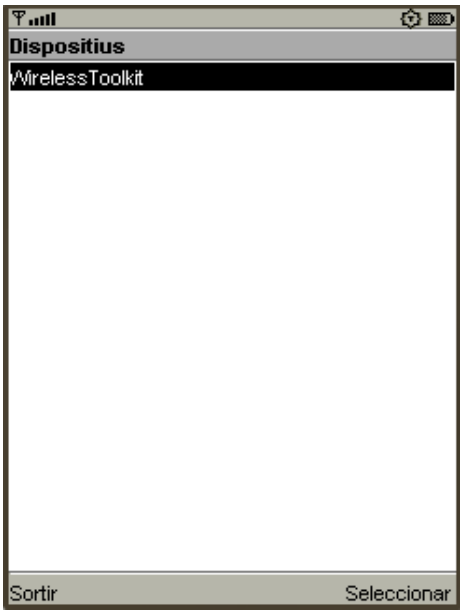

FIGURA 23: Llistat dels dispositius trobats

L'usuari ha d'escollir el dispositiu que representi el Bluetooth de la bicicleta. En aquesta imatge "WirelessToolkit". En acceptar, depenent del dispositiu mòbil, es mostrarà per pantalla un avís de creació de la connexió Bluetooth. Si l'usuari accepta, el telèfon mòbil es connectarà amb la bicicleta.

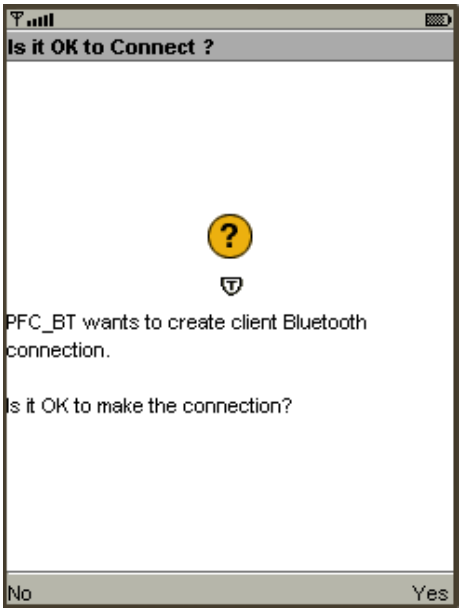

FIGURA 24: Pantalla d'alerta al establir la comunicació

Un cop connectat a la bicicleta, l'aplicació demanarà a l'usuari on vol desar el fitxer de log que registrarà les dades. Per això, s'obrirà un navegador d'arxius amb l'opció "Guardar". En el navegador d'arxius es mostren les carpetes i l'usuari pot seleccionar-les amb l'opció "Seleccionar". Es recomana crear una carpeta per guardar allà tots els arxius de log generats. Depenent del tipus de telèfon mòbil, sortirà un avís a l'usuari indicant que l'aplicació llegirà els seus arxius:

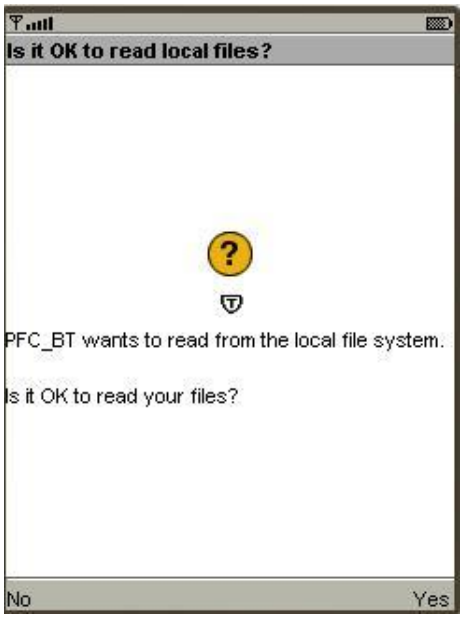

FIGURA 25: Pantalla d'alerta al llegir arxius

En acceptar, es mostraran les carpetes i arxius continguts:

FIGURA 26: Navegador d'arxius

A l'escollir l'opció "Guardar", l'aplicació crearà un arxiu amb la data del dia, en format dd-mm-aaaa. Si en un mateix dia es fa més d'un exercici s'afegirà una extensió numèrica creixent. Aquest podria ser un exemple de nom d'arxiu: 11-01-2010 (7).txt

Un cop guardat l'arxiu es procedirà començar de l'activitat per part de l'usuari i la bicicleta començarà a enviar les dades captades pel cardiòmetre i pel sensor de força. Aquestes dades seran mostrades per pantalla de la següent manera:

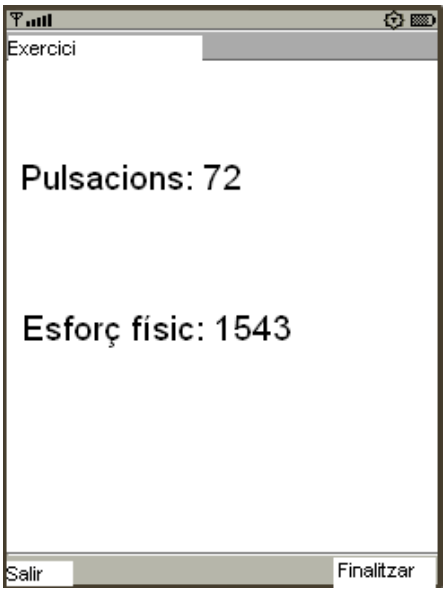

FIGURA 27: Pantalla de seguiment de l'exercici

Quan l'usuari selecciona "Finalitzar" o "Sortir", es tanca la comunicació amb la bicicleta i es mostra el menú inicial. L'usuari ha acabat l'exercici i al seu telèfon mòbil queda el fitxer de log on es guarden totes les dades a la ubicació que ell ha escollit. Un cop finalitzat l'exercici i guardat l'arxiu l'usuari el pot descarregar al seu PC o pot obtenir un resum mitjançant aquest software. Per obtenir aquest resum de l'exercici, triarem al menú inicial l'opció "Detall d'Exercici".

A.3.3 DETALL D'EXERCICI

L'usuari pot obtenir un resum de les dades generades durant un exercici del qual tingui guardat el fitxer de log al seu dispositiu mòbil. Per obtenir aquest resum ha de seleccionar l'opció del menú principal: "Detall d'Exercici".

A l'escollir aquesta opció, es mostrarà de nou el navegador d'arxius, aquest cop per permetre a l'usuari seleccionar l'arxiu de log que vol examinar. Per seleccionar un arxiu s'ha de polsar el botó OK del telèfon mòbil.

Un cop escollit l'arxiu a examinar, l'aplicació llegeix el seu contingut i mostra 3 estadístiques bàsiques del paràmetre pulsació i esforç: La mitjana aritmètica, el valor màxim i el valor mínim. Primer amb una pantalla dedicada a les pulsacions només:

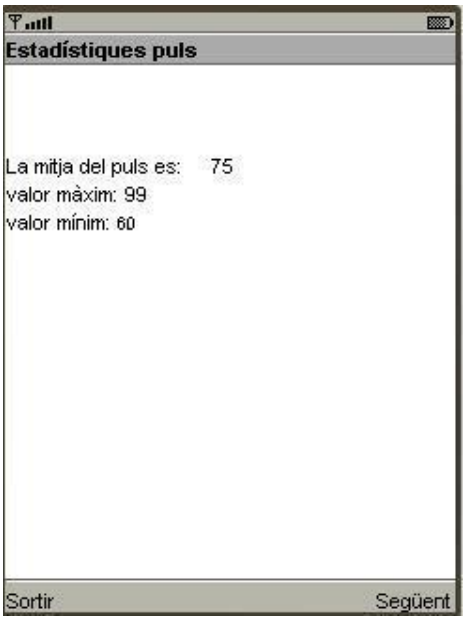

FIGURA 28: Pantalla d'estadístiques

En polsar "Següent" apareixerà la pantalla de les estadístiques de l'esforç:

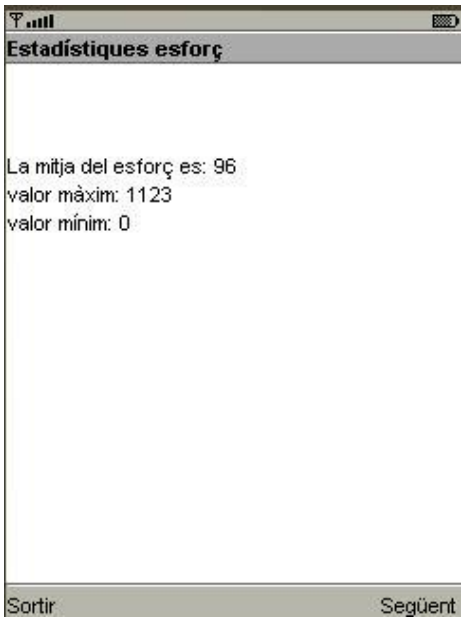

FIGURA 29: Pantalla d'estadístiques

Un altre cop en polsar l'opció "Següent" avançarem en les estadístiques i sortirà una gràfica que mostra la progressió en el decurs de l'exercici tant de les pulsacions com de l'esforç realitzat per part de l'usuari:

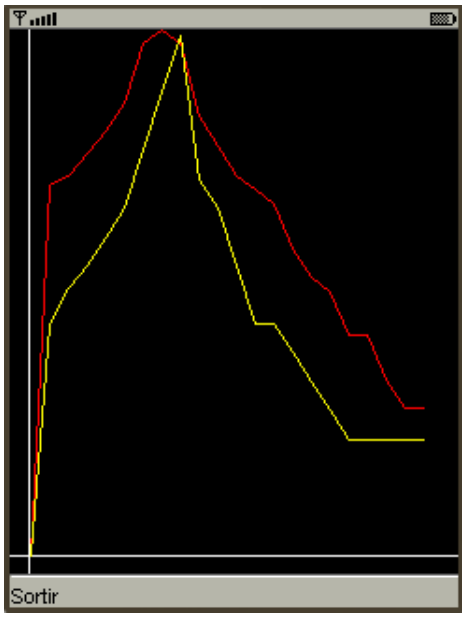

FIGURA 30: Gràfic resultant de l'exercici

En aquesta gràfica que es pot veure, tenim l'esforç en color groc i les pulsacions en color vermell. Aquesta gràfica permet a l'usuari fer-se una idea de l'evolució al llarg de l'exercici dels paràmetres estudiats. En aquesta pantalla ja no es dóna l'opció de continuar, només es pot "sortir" cap al menú inicial.

Tant a les pantalles de les estadístiques com a la pantalla final del gràfic s'ofereix la possibilitat de sortir en tot moment cap al menú inicial.

ANNEX B

LA PLACA DE DESENVOLUPAMENT

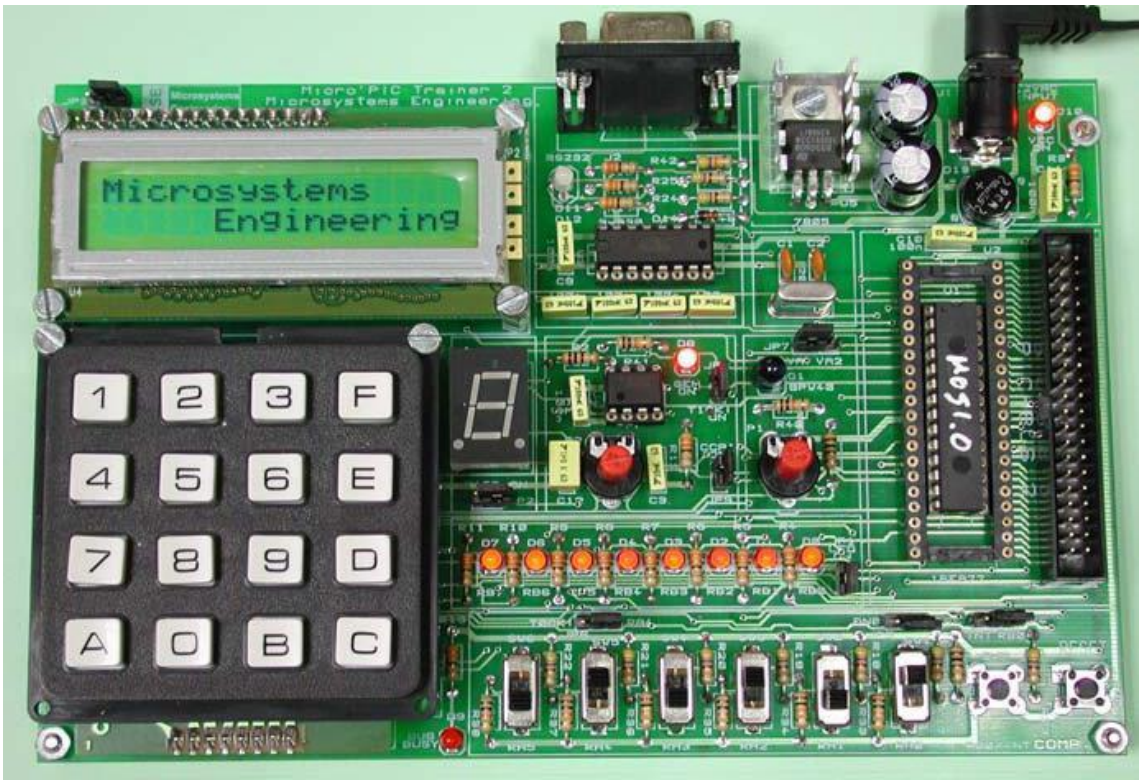

FIGURA 31: Placa de desenvolupament

Per desenvolupar el projecte ens hem ajudat de la placa de desenvolupament PIC Laboratory de l'assignatura de la FIB SDMI. Aquesta placa és utilitzada pels estudiants per desenvolupar pràctiques utilitzant els diferents elements que conté. En aquest projecte la farem servir per emular la bicicleta elèctrica, generar les dades, captar-les i transmetre-les a través del canal sèrie cap al mòbil mitjançant l'adaptador Bluetooth. A continuació tenim una breu explicació dels elements que component aquesta placa.

B.1 LA FONT D'ALIMENTACIÓ

La placa obté la seva tensió de treball de +5Vcc per si mateixa a partir de la tensió d'entrada de 12VAC o de +12VDC. Amb aquesta tensió s'alimenten tots els elements que formen la placa, des de la pantalla lcd fins al microcrontrolador.

FIGURA 32: Font d'alimentació

B.2 EL MICROCONTROLADOR

El microcontrolador és el PIC16F876 de Arizona Microchip, encara que la placa pot suportar el PIC16F477. Aquest microcontrolador que pertany a la gama mitja és molt potent, flexible i conté molts recursos que, alguns d'ells, farem servir per al projecte. Ens proporciona sobradament tot lo que necessitem per desenvolupar el projecte. Ve gravat de fàbrica amb el Sistema Operatiu PICMOS que facilita diverses accions com per exemple carregar un nou firmware.

FIGURA 33: Microcontrolador

B.3 L'OSCIL·LADOR

L'oscil·lador de la placa consta d'un cristall de quars de 20MHz i dos condensadors. Així s'aconsegueix una velocitat de treball de 20MHz que comporta un temps de cicle de 50nS. Com que cada instrucció necessita 4 cicles de rellotge tenim que el temps de cicle d'instrucció és de 200nS. Les instruccions de salt o de desplaçament del comptador de programa (PC) necessiten de dos cicles d'instrucció per executar-se. Aquesta alta velocitat ens permet desenvolupar aplicacions d'alt rendiment i poder obtenir el valor dels sensors amb molta rapidesa. No obstant, no necessitem les dades a cada instant, i les mesures es prenen amb un cert marge de temps entre elles.

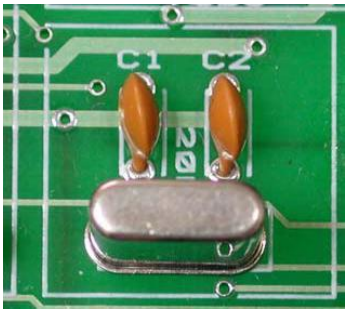

FIGURA 34: Oscil·lador

B.4 LES ENTRADES DIGITALS

La placa disposa de sis entrades digitals (commutadors) i dos polsadors SW1 – SW8 associats a les línies RA0-RA5. No obstant, a la línia RA0 l'usuari pot escollir entre aquestes entrades digitals o l'entrada analògica. També succeeix el mateix amb l'entrada RA4 que es pot seleccionar com a entrada provinent del generador de polsos. El polsador SW7 es pot connectar a l'entrada RB0/INT del microcontrolador per provocar una interrupció externa INT. Per la seva banda, el polsador SW8 provoca un RESET general del sistema.

FIGURA 35: Entrades digitals

B.5 ELS GENERADORS ANALÒGICS

Per aprofitar el convertidor AD del microcontrolador, la placa disposa de dos generadors de tensió variable seleccionables mitjançant un *jumper* com es pot observar a la figura número 36. Un és un foto transistor BPW40 que mesura la llum ambient al voltant de la placa i genera una tensió proporcional. L'altre és un potenciòmetre que proporciona una tensió de sortida proporcional al recorregut del seu potenciòmetre. Aquest últim és el que farem servir per tal d'emular l'esforç que realitza l'usuari mentre realitza un exercici.

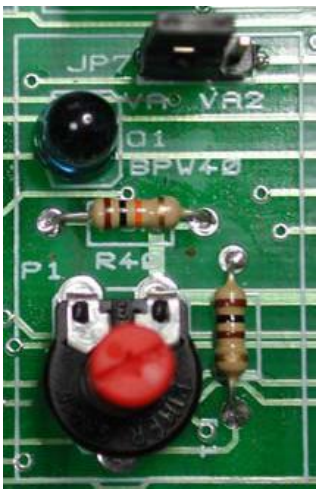

FIGURA 36: Generadors analògics

B.6 EL GENERADOR LÒGIC

Un altre dels elements importants és el generador lògic. Aquest generador es pot fer servir per generar una senyal externa al microcontrolador i utilitzar els seus mòduls de captura i comparació o comptadors. Aquests generador proporciona una ona quadrada asimètrica de freqüència ajustable amb el potenciòmetre i que va de 1Hz fins a una mica més de 150Hz que aprofitarem per emular les pulsacions de l'usuari a l'hora de realitzar un exercici a sobre de la bicicleta. Al costat del potenciòmetre hi ha un led que ens dóna una idea visual de la freqüència de sortida.

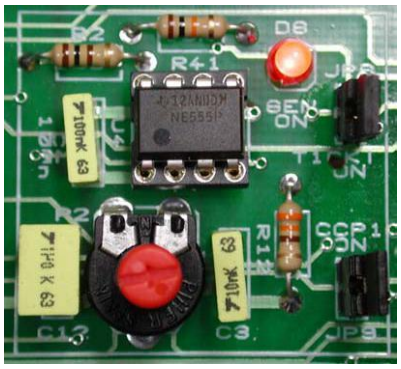

FIGURA 37: Generador lògic

B.7 EL TECLAT

La placa té un teclat matricial 4 x 4 que permet l'entrada de dades. La sortida està connectada a les 8 línies del PortB del microcontrolador. L'objectiu del projecte no és que l'usuari entri cap dada, per això no farem servir aquest perifèric. El control de l'exercici per part de l'usuari es realitzarà mitjançant el telèfon mòbil. Al PortB també tenim associades 8 sortides digitals. Es poden representar per un display de 7 segments que permet la visualització de valors numèrics o per un conjunt de 8 leds. Tampoc farem servir cap d'aquestes sortides.

FIGURA 38: Teclat

B.8 SORTIDES DIGITALS

Estan formades per un conjunt de 8 díodes de tipus led que serveixen per representar l'estat lògic de les línies a les que estan connectats RB0-RB7, que són línees de propòsit general. Les línees de sortida del PIC poden activar directament sense circuit addicional cargues de fins a 25mA. D'aquesta manera aquests leds estan connectats directament afegint només les resistències d'absorció i un nivell lògic "1" encén el led i un nivell "0" l'apaga. Per activar o desactivar aquestes sortides només hem de manipular el *jumper* 1. A més a més, la línia RB0

associada al primer led es pot utilitzar com a entrada de la interrupció externa d'un polsador situat a l'extrem inferior de la placa de desenvolupament.

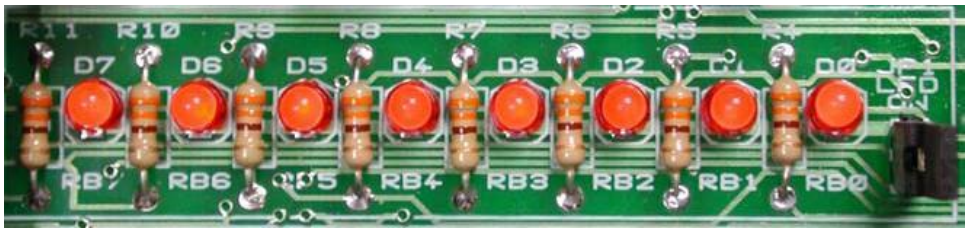

FIGURA 39: Sortides digitals

B.9 SORTIDA A DISPLAY DE 7 SEGMENTS

Un altre perifèric de la placa de desenvolupament és un clàssic display led de 7 segments que pot mostrar un valor numèric. A efectes pràctics cada segment és com un led connectat a la sortida del port B del microcontrolador. S'afegeix també una resistència com l'esmentada a l'apartat anterior i el funcionament és igual, nivell "1" encès, nivell "0" apagat. Mitjançant el *jumper* J2 es pot desconnectar aquest element evitant el consum sobre les línies RB0-RB7.

FIGURA 40: Display de 7 segments

B.10 LA PANTALLA LCD

Un dels elements que més salta a la vista és la pantalla LCD de doble línia amb 16 caràcters alfanumèrics per línia. Aquest mòdul es connecta a les línees RB0 - RB7 per fer les transferències dels caràcters en codi ASCII, que es fa en conjunts de 4 o 8 bits. La pantalla

LCD rep la tensió d'alimentació +5Vcc des de la font d'alimentació i així alimenta tota l'electrònica interna. Per desconnectar la pantalla només cal fer servir el *jumper* 3.

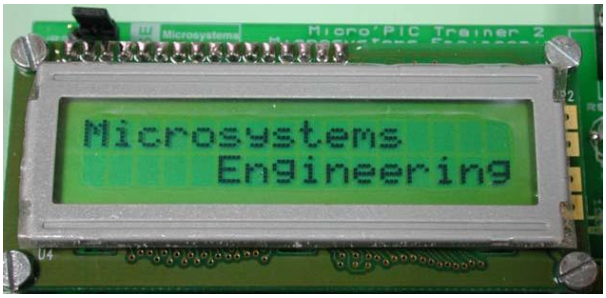

FIGURA 41: LCD

B.11 EL CANAL SÈRIE RS-232

Un dels elements essencials per al projecte és el canal sèrie RS232 que incorpora la placa. Per aquest canal carreguem el nostre codi al PIC i ens comuniquem amb l'adaptador Bluetooth

passant-li les dades captades. El circuit de la placa per adaptar els nivells lògics a físics és el popular MAX232. Aquest circuit pot obtenir nivells +/- 12V a partir de nivells lògics TTL i a l'inrevés.

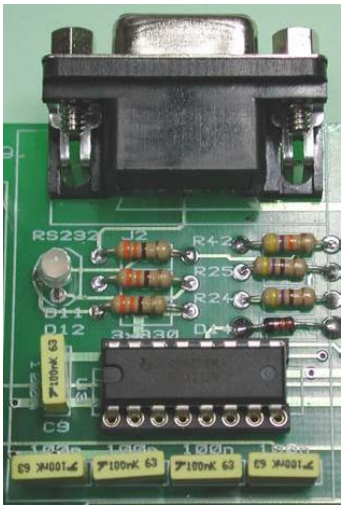

FIGURA 42: Canal sèrie

ANNEX C

ADAPTADOR BLUETOOTH – RS-232

C.1 DESCRIPCIÓ GENERAL

L'empresa Parani-SD ha creat aquest terminal per adaptar el canal sèrie a la comunicació sense fils fent servir la tecnologia Bluetooth, un dels principals estàndards wireless de curt abast. Aquest adaptador es pot comunicar amb altres dispositius Bluetooth, però han de suportar el protocol btspp, Bluetooth Serial Port Profile.

El model en concret utilitzat per aquest projecte és el Parani-SD 200, que té un abast de 30 metres.

Un dels avantatges d'aquest adaptador és el seu disseny compacte que facilita l'emplaçament al ser connectat a un altre equip. Aquest podria ser un factor important a l'hora d'implementar el projecte a la bicicleta. Per aquest projecte, no obstant, no és determinant ja que connectarem l'adaptador Bluetooth a la placa de desenvolupament. Disposa d'una antena extraïble per optimitzar la qualitat i la distància de la comunicació.

El Parani-SD suporta FHHS[27], autenticació i encriptament de les dades.

Aquest dispositiu pot ser configurat de diverses formes. Es pot configurar utilitzant comandes AT, software del fabricant ParaniWIN i a través d'uns interruptors a un petit panell de configuració sota el dispositiu. Aquesta darrera configuració, però, no pot configurar tants paràmetres com les altres dues opcions.

C.2 ESPECIFICACIONS TÈCNIQUES DEL PRODUCTE

A continuació hi ha una taula amb les especificacions tècniques del producte:

Nom	Parani-SD 200
Interfície Sèrie	Comunicació de dades pel connector DB9 femella Velocitat de transferència: 1200bps fins 230400bps Control de flux: Cap/Hardware Senyals: Rx, Tx, RTS, CTS, DTR, DSR, GND, DCD
Interfície Bluetooth	Bluetooth v1.2 * Protocol: RFCOMM, L2CAP, SDP Profile: Serial Port Profile
	Classe 2 Nivell: Max. 4dBm Distància de treball amb l'antena per defecte: 30m
Configuració	Software fabricant (ParaniWIN), Comandes AT i Hardware
Actualització firmware	ParaniUpdater
LED de diagnòstic	Encès Standby Connectat Comunicació sèrie Rx/Tx
Alimentació	Tensió: 5V ~ 12V DC Potència Nominal Consumida: 58mA@9600bps, 66mA@115Kbps
	Alimentació a través de: Via alimentador estàndard connector-AC adaptador-DC Cable d'alimentació USB Cable DC A través del Pin 9 del connector DB9 Bateria dissenyada pel fabricant
Temps de bateria	Fent servir les bateries subministrades pel fabricant 1.5V AA piles alcalina (2500mAh): Mitjana: 11hr@9600bps
Ambient	Temperatura òptima: -10 ~ 55º C Temperatura suportada: -20 ~ 70º C 90% d'humitat (no condensada)
Mides	101 mm Llarg (4.0 in.) 31 mm Profund (1.2 in.) 16 mm Alt (0.6 in.)

Pes	23 g

C.3 VISTA DEL DISPOSITIU

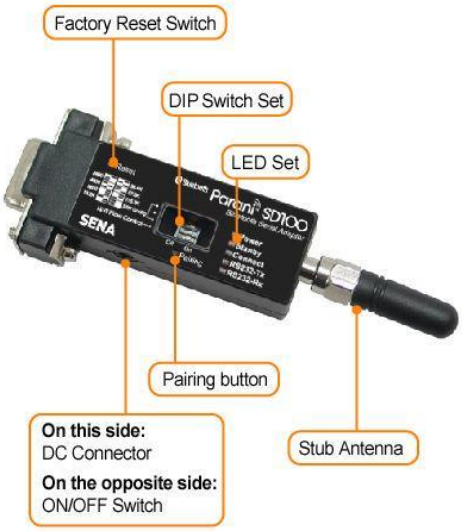

FIGURA 43: Imatge del PARANI-SD 200

Com es pot apreciar a la imatge, el Parani-SD 200 incorpora l'antena abans esmentada per millorar la qualitat i l'abast de la comunicació. Disposa d'un conjunt d'interruptors per configurar els paràmetres de comunicació més bàsics. A més també disposa d'un conjunt de LED que indiquen diversos estats. Tots aquests estats estan indicats al quadre d'especificacions tècniques del producte. El Parani-SD disposa de dos botons; el de reinici i el de connectar-se a un altre dispositiu Bluetooth Parani-SD, anomenat Pairing Button. Amb aquest botó no es necessita d'un PC addicional per realitzar la connexió entre els dos aparells.

Als extrems de la capsa que conté el chip està l'interruptor d'encès i el connector de tipus jack per a l'alimentació

C.4 CONNECTANT EL DISPOSITIU

En aquesta secció es mostra les diferents maneres de connectar el dispositiu a l'alimentació i a un altre dispositiu sèrie.

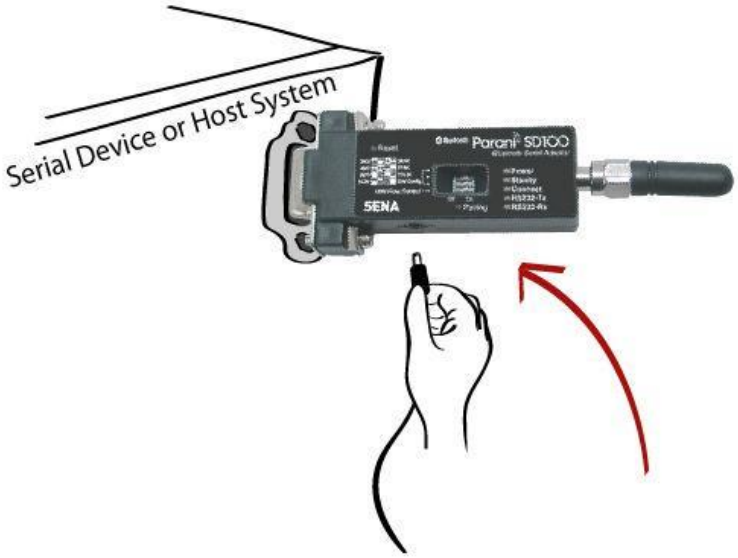

FIGURA 44: Com connectar a la alimentació

En aquesta figura es mostra com, ja connectat a un altre dispositiu, es pot connectar el cable d'alimentació per la part de sota. El connector és de tipus jack i es connecta fent servir l'adaptador d'alimentació DC inclòs al paquet del producte. En connectar-se s'il·lumina el LED indicador de la connexió a la alimentació.

Per connectar el dispositiu Parani-SD a un altre dispositiu només cal que l'altre dispositiu tingui un connector DB9 mascle. Si s'estableix comunicació a través del canal sèrie els LED de transmissió sèrie Rx/Tx s'encendran per mostrar aquesta activitat. Per connectar el Parani-SD a un altre dispositiu es procedirà de la manera següent:

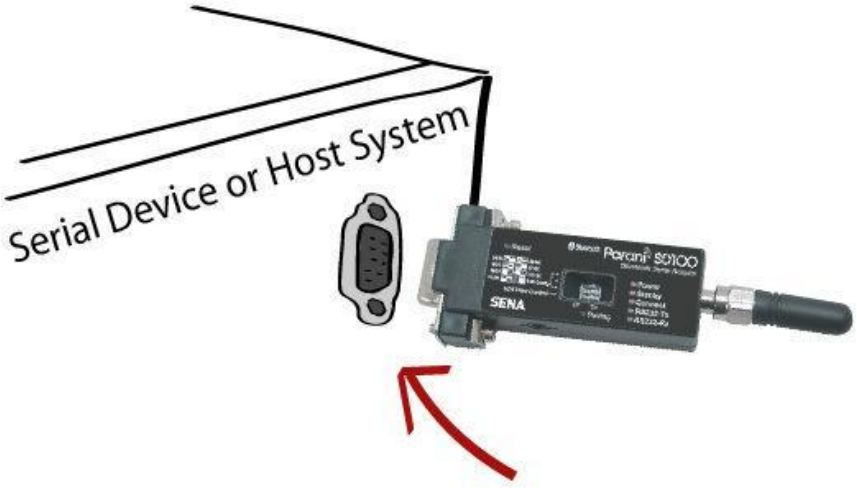

FIGURA 45: Connexió a un altre dispositiu

C.5 INDICADORS LED

Com s'ha esmentat en diverses ocasions prèviament, el Parani-SD 200 disposa de 4 LED indicadors. Aquests LED emeten llum en segons què circumstàncies. Depenen del mode d'operació en el que estigui l'aparell s'encendran seguint aquests patrons:

Indicator	Power LED	Standby LED	Connect LED
Mode0	Green ┌────	Red ┌────	
Mode1	Green ┌────		Green (every 1 sec) ┌┐
Mode2	Green ┌────		Green (every 3 sec) ┌┐┌┐
Mode3	Green ┌────		Green (every 3 sec) ┌┐┌┐
Connected	Green ┌────		Green ┌────

FIGURA 46: Taula indicadora del significat dels LEDS

El LED que mostra si està connectat es manté sempre encès si hi ha algun dispositiu connectat al Parani-SD 200. El mateix succeeix amb el LED indicador de l'alimentació, roman encès mentre estigui connectat a la xarxa.

C.6 CONNEXIÓ A LA BATERIA

Per aquest projecte no resulta imprescindible la connexió a les bateries ja que l'objectiu és emular la bicicleta amb la placa de desenvolupament al laboratori. No obstant, per a la implementació real a la bicicleta elèctrica s'hauria de tenir en compte.

El dispositiu Parani-SD 200 que es fa servir suporta una bateria que porta dues piles AA. Aquestes piles poden ser estàndards o recarregables, encara que si són d'aquestes últimes el pack de la bateria subministrada amb el producte no les recarrega per si mateix.

FIGURA 47: Connexió a la bateria

C.7 BOTÓ "PAIRING"

Els dispositius Parani-SD 100 i 200 disposen de botó per emparellar-se de forma automàtica i sense PC de manera instantània. Per connectar dos dispositius es procedirà de la següent manera:

1. Apagar tots els Parani-SD propers.
2. Encendre els dos dispositius que es volen connectar i pressionar el botó per resetejar el hardware en els dos.
3. Pressionar el botó d'emparellar al dispositiu 1 durant dos segons fins que s'apagui el LED de standby i fins que pampallugui 3 cops el bit de connexió.
4. Fer la mateixa operació anterior amb l'altre dispositiu.
5. Esperar fins que es connectin entre ells els dos dispositius, que serà quan tots dos mostrin el LED connexió encès permanentment.
6. Apagar i encendre el dispositiu 1. Al encendre's de nou, el LED connexió s'encendrà 2 cops cada 3 segons.
7. Apagar i encendre el dispositiu 2. Al encendre's de nou, el LED connexió s'encendrà 1 cop cada segon.

Ara els dos dispositius estan configurats per connectar-se de manera automàtica quan estiguin encesos.

ANNEX D

GLOSSARI

[1] Bluetooth: Estàndard de comunicacions sense fils per l'intercanvi de dades a curta distància.

[2] Firmware: Nom per designar els programes i les estructures de dades que internament controlen els aparells electrònics.

[3] RS-232: (Recommended Standard 232) és un estàndard per a la transmissió sèrie de dades entre dispositius.

[4] IEEE: (Institute of Electrical and Electronics Engineers) és una organització internacional per al desenvolupament de la tecnologia relacionada amb l'electricitat.

[5] JAVA Micro Edition: és una plataforma JAVA dissenyada per a dispositius mòbils i sistemes encastats.

[6] API: application programming interface, abstracció que defineix una interfície per a la interacció amb el seu conjunt de funcionalitats.

[7] URL: Uniform Resource Locator, especifica la localització d'un servei.

[8] MIDlet: és una aplicació JAVA per a dispositius mòbils o altres sistemes encastats.

[9] Listener: funció que està a l'espera d'un event en concret.

[10] CLASSPATH: és una variable d'entorn que indica a la Java Virtual Machine on buscar les classes i els paquets de JAVA.

[11] CLDC: Connected Limited Device Configuration, és una especificació del marc de treball per a les aplicacions de JAVA ME.

[12] MIDP: Mobile Information Device Profile, és una versió de JAVA ME integrada en els sistemes encastats com dispositius mòbils.

[13] Stream: flux de dades

[14] Baud rate: és una magnitud que mesura el nombre mig de senyals transmesos per segon.

[14] Plugin: és una extensió d'un programa principal que interactua amb aquest per proveir-lo d'una determinada funció o servei.

[16] Jumper: component electrònic que s'utilitza per obrir o tancar una part d'un circuit elèctric.

[17] Estat Sleep: estat d'un microcontrolador en que es deshabiliten diversos elements i es canvia la configuració per reduir el consum d'energia

[18] Timer 555: El Timer 555 és un circuit integrat dissenyat per Hans R. al 1970 molt utilitzat degut al seu senzill ús, baix preu i bona estabilitat.

[19] Preescaler: Preescaler és un component electrònic que es fa servir per reduir la freqüència d'un senyal elèctric mitjançant una divisió entera.

[20] USART: Universal Synchronous Asynchronous receiver transmiter.

[21] Full duplex: en comunicació, quan els dos dispositius poden enviar i rebre dades al mateix temps.

[22] Bootloader: és un programa que carrega una aplicació al microcontrolador.

[23] Dispositiu sèrie DCE: Data Communication Equipment, és un dispositiu de comunicació entre el DTE, Data Terminal Equipment i la xarxa de telecomunicació. Un exemple podria ser el PC com a DTE i un mòdem com a DCE. Enmig residiria la xarxa telefònica.

[24] IDE: són les sigles de integrated development enviroment, que és un software especialment dissenyat per al desenvolupament d'aplicacions que integra un editor de codi, compilador i debugger.

[25] Threads: és una seqüència d'instruccions que s'executen en paral·lel amb altres Threads.

[26] Deadlock: Un deadlock o abraçada de la mort és quan un procés espera a un altre que mai acabarà.

[27] FHHS: Frequency Hopping Spread Spectrum, que és una tècnica que permet al Parani-SD minimitzar les interferències de radio.

AGRAÏMENTS

M’agradaria, des d’aquestes darreres línies de la memòria del projecte, donar-li les gràcies a totes les persones que l’han fet possible.

Al meu director, Joan Climent, pel recolzament, orientació i consells que m’ha donat durant tot el decurs del projecte.

A la meva parella, Thais Diego, per tota la seva ajuda en aspectes com la correcció de faltes ortogràfiques i gramaticals i pel magnífic disseny de la portada de la memòria.

A la meva família per l’ajuda econòmica en els moments més difícils durant la realització del projecte.

A tots els professors de la facultat que he tingut durant aquests quatre anys i escaig.

I finalment als companys de classe que m’han ajudat en aquests anys i que avui dia són més que companys, són amics.

Gràcies a tots,

Jose Carlos Yáñez Nieto